台北富邦銀行 Taipei Fubon Bank

鷹眼識詐聯盟技術交流

數據科學部 劉晉儒 2023.12.19

資料欄位問題

No	欄位名稱	問題	回答
1	AUM(NTD, 萬元)	1.是否僅計算台幣資產(支、活、定)即可，需要包含外幣、放款、理財商品等嗎？ 2. AUM AMT欄位若初期資料僅納入台外幣存款支存+存款定存+存款活存，未包含(信託其他+信託指單+海外股票+國外基金+國內基金+國內基金代銷+組合式商品+債券+債券附條件+黃金存摺等資料)，若後續增加如信託、基金、黃金資料，是否會影響模型訓練成效?	1.AUM_AMT請盡可能包含台外幣存款支存+存款定存+存款活存+信託其他+信託指單+海外股票+國外基金+國內基金+國內基金代銷+組合式商品+債券+債券附條件+黃金存摺，為現在市值折台幣庫存餘額，本行存款、投資各自用不同時間點的匯率進行折台，折台會率請依各自行庫規定即可。. 2.請盡可能含蓋財管商品，惟AUM非最重要特徵，資料候補不嚴重影響成效。
2	CHANNEL_DESC通路說明	1.路說明及通路兩欄位想請教貴行是否能提供分類的依據及邏輯？另外，貴行通路說明只篩選行銀、網銀及自行ATM三種？如果通路為臨櫃，那其通路說明為何？	與通路CHAL_1不同，CHAL_1區分為ATM、Online、Mobile、Branch、Batch，通路說明無須與富邦代碼對齊，有以下分類即可:網銀、行銀、自行ATM、跨行ATM、銀資交換匯款、一卡通(儲值)、端末(臨櫃)
3	EMP_NO櫃員編號	1. .請問櫃員編號指的是什麼? 是指交易的通路嗎? 若是，與【活存交易檔】之編號「29通路說明及30通路」的差別？	櫃員編號目的為限縮交易範圍，通路說明為交易通路區分，此欄位較適合用作通路特徵建立。 富邦代碼如下(01、02、03為不同路徑)，需有以下分類，無須與富邦代碼對齊 4001:跨行提款 4002:跨行轉帳 4005:跨行匯入 4011:網路銀行櫃員01 4012:網路銀行櫃員02 4013:網路銀行櫃員03 4014:行動銀行 4021:晶片卡跨行提款 4022:晶片卡跨行轉帳 4111:行動銀行櫃員01 4112:行動銀行櫃員02 4115:網路銀行櫃員 4215:新行動銀行 3開頭:自動櫃員機

資料欄位問題

No	欄位名稱	問題	回答
4	user_id_level/通路說明	1.類別型變數轉dummy時，是否有處理虛擬變數陷阱問題? 2.user_id_level (特徵序號25 約轉設定通路)是否有轉dummy?	1.因為非回歸問題，亦非使用線性模型，虛擬變數陷阱確實造成運算資源消耗，但對樹結構模型影響不大。 2.user_id_level在進行特徵工程用以計算通路數。
5	券商集保代碼	因券商集保代號(欄位16)難拿到，若本欄空值是否影響模型運作？	券商授權檔-券商集保代號並非建模特徵，而是建模前用以排除客群的條件。如未有此欄位，可直接找出證券互註記即可。
6	環境問題	1. 模型執行環境的硬體規格(cpu個數、記憶體容量、硬碟容量、GPU版本、作業系統版本等等)，能否提供完整的主機規格？ 2. 為避免python3.6版本EOS問題，模型執行環境若使用python3.9版本能否順利執行？是否會衍生出其他風險及需要注意的地方？麻煩提供一版對應python3.9的套件版本清單	1.建議使用4CPU、32GB記憶體容量或8CPU、64GB記憶體(以上)，請視自行資料量而定。GPU的規格型號為NVIDIA RTX 2080Ti、Hadoop作業系統Red Hat Enterprise Linux 7.9、Hadoop版本 : CDP 7.1.7 2.目前行內因無Python3.9環境，建議遇到版本不同套件至Pypi網站下載

AGENDA

01 程式碼架構

02 存款交易特徵工程

03 網行銀軌跡特徵工程(流程供參)

程式碼架構

程式碼

Job_SQL語法

POLICE_DASB_job_cust_pred.sql
POLICE_DASB_salary_acct.sql
POLICE_DASB_job_tracking_log_pred.sql

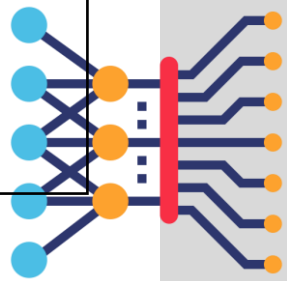
Job 排程腳本

01.data_cleaning.py
02.feature_engineering_1.py
03.feature_engineering_2.py

Utils函式

config_tbl.py
data_cleansing.py
feature_engineering_2023.py
model_testing.py

Model模型



xgb_model_2022-08-31_fin_v1
second_xgb_model_2022-08-31_fin_v1

存款交易特徵工程

程式碼

Job_SQL語法

POLICE_DASB_job_cust_pred.sql
POLICE_DASB_salary_acct.sql
POLICE_DASB_job_tracking_log_pred.sql

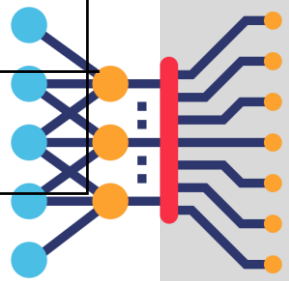
Job 排程腳本

01.data_cleaning.py
02.feature_engineering_1.py
03.feature_engineering_2.py

Utils函式

config_tbl.py
data_cleansing.py
feature_engineering_2023.py
model_testing.py

Model模型



xgb_model_2022-08-31_fin_v1
second_xgb_model_2022-08-31_fin_v1

特徵工程-1

確認交易資料

nonwarning.csv

acct_nbr	act_date	tx_date	tx_time	drcr	cust_id	own_trans_acct	tx_amt	pb_bal	tx_brh	cur	channel_desc	chal_1	memo	remk	jrnl_no	tx_mode
28635900	2023-04-10 08:00:00	2023-04-10	2023-04-10 12:22:10	1 A	7 00	185400	8750.0	0.0	00851	TWD	網路銀行 行銀	Online	行動自轉 00	5400	015562830	1
28635900	2023-04-10 08:00:00	2023-04-10	2023-04-10 11:17:02	1 A	7	None	50000.0	0.0	00817	TWD	銀行資訊交換平台FEP自行ATM	ATM	CD提款	0716	014427418	1
40552600	2023-04-12 08:00:00	2023-04-12	2023-04-12 13:32:34	1 A	7	None	2000.0	10558.0	00851	TWD	網路銀行 行銀	Online	行動跨轉 0001	4900	015014130	1
47996500	2023-04-19 08:00:00	2023-04-19	2023-04-19 12:44:05	1 F	2 00	155900	64.0	0.0	00301	TWD	銀行資訊交換平台FEP自行ATM	ATM	CD轉帳 0000	5900	014395487	1
73525100	2023-04-06 08:00:00	2023-04-06	2023-04-06 17:07:14	2 F	2	None	20000.0	20017.0	011	TWD	銀行資訊交換平台FEP跨行ATM	Online	CD轉收 0061	3624	021877889	1

acct_nbr_ori	acct_nbr	act_date	tx_date	tx_time	drcr	cust_id	own_trans	tx_amt	pb_bal	tx_brh	cur	channel_desc	chal_1	memo	remk	jrnl_no	tx_mode
母帳號	子帳號	帳務日	交易日	交易時間	轉入出	統編	對手帳號	金額	餘額	分行	幣別	通路說明	通路	摘要	備註	序號	交易狀況

特徵工程-1

特徵工程腳本

02.feature_engineering_1.py

呼叫函式 import utils.feature_engineering_2023 as fe

```
##############
#### 指引 ####
##############
# 請依下方TODO修改路徑
# Step1: 修改腳本directory (TODO_1)
# Step2: 確認自己富邦utils的functions要放在哪裡 (TODO_2)
# Step3: 在TODO_1的directory下新增Log資料夾 (TODO_3)
# Step4: 請修改config_tbl.py中VIEW及表名，如無spark可忽略 (TODO_4)
# Step5: 需修改Data cleansing後Parquet存放位置(TODO_5)
# Step6: 需修改Feature Engineering後存放位置(TODO_6)

#####################################
#### 更改當下路徑到運行腳本的資料夾路徑 ####
#####################################
import os
# 將工作路徑改到此 script 的路徑
abspath = os.getcwd()
print(f'Original absolute path: {abspath}')
print('Change working directory...')
os.chdir(r'./P&B_Development/Training')                # TODO_1: 因為之後要使用富邦utils的functions，所以一定要改目錄到相應的位置
abspath = os.getcwd()
print(f'Current absolute path: {abspath}')


####################
## Import packages ##
####################
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'
import json
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score,f1_score,confusion_matrix,make_scorer, accuracy_score, precision_score, recall_score
import pickle
from datetime import date, datetime, timedelta
from pyspark.sql.functions import col, countDistinct, when, round, ceil, trim, concat, coalesce, monotonically_increasing_id, datediff, date_format, row_number, to_timestamp, concat,
x_timestamp, lit, collect_list
from pyspark.sql.window import Window
from pyspark.sql import functions as F
import random
from datetime import date, datetime
import dateutil
import logging
from logging.handlers import RotatingFileHandler
import utils.feature_engineering_2023 as fe            # TODO_2: 確認自己富邦utils的functions要放在哪裡，決定TODO_1位置
from IPython.display import display
import utils.config_tbl as config
```

特徵工程-1

特徵工程函示庫

utils.feature_engineering_2023.py

進行特徵工程過程可忽略由spark取得來源表的過程

```
import os
# 新增環境變數(一定要加的)
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

## Spark Utils
#from pyspark.sql import SparkSession, HiveContext
#from pyspark import SparkContext, SparkConf, SparkFiles
#from pyspark.sql.types import StructType, StructField, TimestampType,StringType, IntegerType, FloatType, BooleanType, LongType, DateType
#from Job.config import psid_info
#from pyspark.sql.functions import col, countDistinct, when, round, coalesce, monotonically_increasing_id, date_format, row_number, to_timestamp, concat, unix_timestamp, lit,collect_list, to_date,
#
## relative to spark
#import pyspark.sql.functions as pys
#from pyspark.sql.window import Window
#from pyspark.sql import functions as F
#from pyspark.sql import Row

# time utils
import time
from datetime import date, datetime, timedelta
import dateutil.relativedelta

# Dataframe
import pandas as pd
import numpy as np
import pickle

# other packages
import json
import pprint
```

特徵工程-1

特徵工程函示庫

utils.feature_engineering_2023.py

Job 排程腳本

01.data_cleaning.py
02.feature_engineering_1.py
03.feature_engineering_2.py

Utils函式

config_tbl.py
data_cleansing.py
feature_engineering_2023.py
model_testing.py

fe.preprocessing_source

refer to

daily_total_func

tx_brh_cnt_func

session_drcr_cnt_func

flow_func

cfpebtrfin_func

特徵工程-1

特徵工程腳本

02.feature_engineering_1.py

```
############################
######## 排程特徵工程 ########
############################

try:
    # add logging
    logger.info(f'{"=" * 20} Normal Account Feature Engineering {"=" * 20}')
    logger.info('Normal Account Feature Engineering started')
    start = time.time()
    # 讀取預測交易資料
    nonwarning_spark = pd.read_csv(f'nonwarning.csv')                    # TODO_5:需修改Data cleansing後Parquet存放位置
    # 建立id_list
    nonwarning_id_list = nonwarning_spark['cust_id'].unique().tolist()
    # 取得min、max date
    normal_min_date = nonwarning_spark['tx_date'].min()
    normal_max_date = nonwarning_spark['tx_date'].max()
    #撈取存款主檔資料
    #====
    date = datetime.now()
    date = date - dateutil.relativedelta.relativedelta(days=1)
    date = date.strftime('%Y-%m-%d')
    query = f""" select ACCT_NO_14, ACCT_OPEN_DT from {ACCT_VIEW}.{INVM} where snap_date = '{date}' """
    invm = spark.sql(query).toPandas()
    #===
    #撈取約轉資料
    #====
    query = f''' select * from {HIVE_VIEW}.{TRFIN} '''
    cfpebtrfin = spark.sql(query).toPandas()
    #===
    # 特徵工程
    nonwarning_spark = fe.preprocessing_source(nonwarning_spark, invm = invm, cfpebtrfin = cfpebtrfin, mode = 'train', id_list = nonwarning_id_list, min_date = normal_min_date,
= normal_max_date, date_list = None, source = 'alarm', label = 0)
    logger.info('Normal Account Feature Engineering done.')
    print('Saving file...')
    nonwarning_spark.to_csv('nonwarning_feature_engineering.csv', index = False)    #TODO_6: 需修改Feature Engineering後存放位置
    print('Saving Done.')
    #add logging
    end = time.time()
    logger.info(f'time consumed: {(end-start)/60} minutes')
    logger.info('Normal Account Feature Engineering and data exporting done.')
    logger.info(f'{"="*70}')
except Exception:
    logging.exception('message')
    print('Normal Account Feature Engineering failed.')
    logging.error('Normal Account Feature Engineering failed.')
    logger.info(f'{"="*70}')
```

進行特徵工程過程可忽略由spark取得來源表的過程

確保表轉換為pandas dataframe

特徵工程函式-1

01.當日總轉出、轉入 → 02.提領分行數 → 03.session → 04.flow

utils.feature_engineering_2023.py

```
def daily_total_func(data):
    """
    功能: 建置當日交易總轉入 daily_total_2 當日交易總轉出 daily_total_1 amt_diff 兩者差額比率
    input: 交易資料Spark Dataframe
    output: 交易資料Spark Dataframe
    """
    print('daily_total_amt')
    start = time.time()
    #分群後新增daily_total並將data獨立出為dataframe
    daily_total = data.groupby(['acct_nbr_ori','tx_date','drcr']).agg({'tx_amt': 'sum'}).reset_index()
    daily_total.columns = ['acct_nbr_ori','tx_date','drcr','daily_total']

    #分別計算 daily_total_1 和 daily_total_2
    daily_total_1 = daily_total[daily_total['drcr'] == 1][['acct_nbr_ori','tx_date','daily_total']].rename(columns = {'daily_total': 'daily_total_1'})
    daily_total_2 = daily_total[daily_total['drcr'] == 2][['acct_nbr_ori','tx_date','daily_total']].rename(columns = {'daily_total': 'daily_total_2'})

    #合併daily_total_1和daily_total_2到原始資料
    data = data.merge(daily_total_1, on = ['acct_nbr_ori','tx_date'], how = 'left')
    data['daily_total_1'] = data['daily_total_1'].fillna(0)
    data = data.merge(daily_total_2, on = ['acct_nbr_ori','tx_date'], how = 'left')
    data['daily_total_2'] = data['daily_total_2'].fillna(0)

    #daily_total_1、2為0者改為1
    data['daily_total_1'] = np.where(data['daily_total_1'] == 0, 1.0, data['daily_total_1'])
    data['daily_total_2'] = np.where(data['daily_total_2'] == 0, 1.0, data['daily_total_2'])

    #計算amt_diff
    data['amt_diff'] = abs((data['daily_total_2'] - data['daily_total_1']))/(data['daily_total_2']/2 + data['daily_total_1']/2)

    end = time.time()-start
    print(end,'seconds')
    return data
```

計算每天轉入/轉出金額

計算每天轉入/轉出金額差占平均交易金額占比

特徵工程函式-1

utils.feature_engineering_2023.py

計算日提領分行

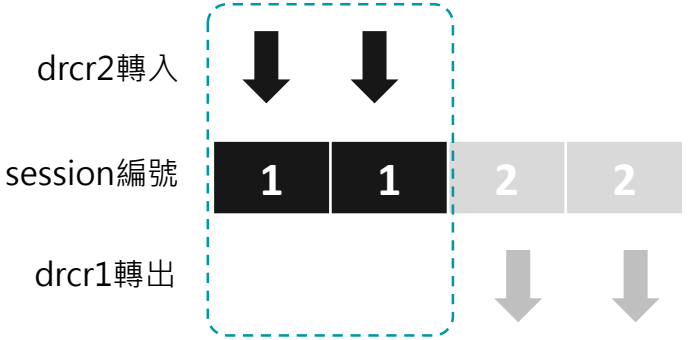
```python
def tx_brh_cnt_func(data):
    """
    功能: 建置日提領分行數
    input: 交易資料Dataframe
    output: 交易資料Dataframe
    """
    print('tx_brh_cnt_func')
    tx_brh_cnt = data.groupby(['acct_nbr_ori','tx_date'])['tx_brh'].unique().reset_index()
    tx_brh_cnt['count'] = tx_brh_cnt['tx_brh'].apply(lambda x: len(x))
    tx_brh_cnt = tx_brh_cnt.drop(['tx_brh'], axis = 1)
    tx_brh_cnt = tx_brh_cnt.rename(columns = {'count': 'tx_brh_cnt'})
    data = data.merge(tx_brh_cnt, on = ['acct_nbr_ori','tx_date'], how='left')
    return data
```

特徵工程函式函式-1

01.當日總轉出、轉入 → 02.提領分行數 → 03.session → 04.flow

utils.feature_engineering_2023.py

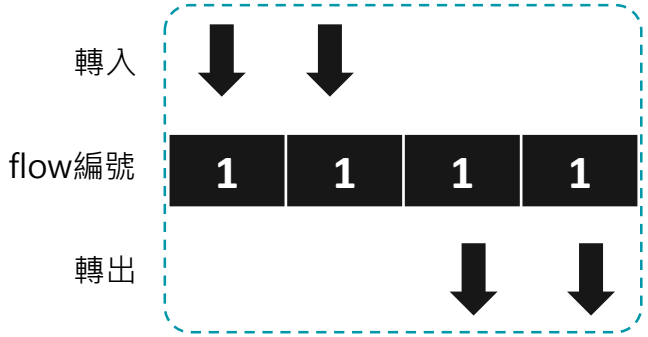
Session: 連續轉出或轉入 (中繼變數)



(衍生)

模型欄位名稱	欄位中文名稱
session_total_amt	session總金額
session_accumulated_amt	session該筆當下累計金額

Flow: 轉入後轉出算同一flow (中繼變數)



(衍生)

模型欄位名稱	欄位中文名稱
flow_total_amt	flow總轉入金額
flow_tx_amt_ratio	flow該筆金額/總轉入金額
flow_tx_amt_seq	flow該筆當下累計金額
flow_tx_amt_seq_ratio	flow該筆當下累計金額佔flow總金額比例
flow_ttl_amt_1	flow總轉出金額
flow_ttl_amt_drcr_ratio	flow總轉出/轉入差額比
flow_avg_time_diff	flow平均交易時間差

特徵工程函式-1

03.session → 04.flow

```python
def session_drcr_cnt_func(data, mode = 'train', source = 'normal', session=0):
    """
    功能: 捕捉連續轉入/出行為，有連續行為則給定session編號
    input: 交易資料Dataframe, mode: 訓練或預測模式, 資料源: 警示戶或正常戶, session: default值0
    output: 交易資料Dataframe
    """
    #在第一個位置插入0
    def insert_lst(x):
        x.insert(0,0)
        return x
    #建立session編號
    def drcr_cnt(x, session):
        # global session 從1開始編號
        session += 1
        # 計算每個session內的交易動作數，從0開始編號
        acct_action_cnt = []
        cnt=0
        session_list = []
        for idx in range(0, len(x)):
            if x[idx]!=0:
                cnt+=1
                acct_action_cnt.append(cnt)
                session_list.append(session)
            else:
                if idx == 0:
                    cnt=0
                    acct_action_cnt.append(cnt)
                    session_list.append(session)
                else:
                    cnt=0
                    session+=1
                    acct_action_cnt.append(cnt)
                    session_list.append(session)
        return acct_action_cnt, session_list
    print('session_drcr_cnt_func')

    if mode == 'train' and source == 'normal':
        session=0
        #groupbykey只針對x.acct_nbr_ori, mapValues(list)針對x.drcr, 類似dict
        #對每個帳號做cnt,判斷連續轉入或轉出,連續時給1
        cnt_list = data.groupby('acct_nbr_ori_1')['drcr'].apply(lambda x: [1 if (x.iloc[idx-1] == x.iloc[idx]) else 0 for idx in range(1, len(x))]).reset_index()
        cnt_list['drcr'] = cnt_list['drcr'].apply(lambda x: insert_lst(x))
        cnt_list[['acct_action_cnt', 'session_list']] = cnt_list['drcr'].apply(lambda x: pd.Series(drcr_cnt(x, session)))
        action_cnt = pd.DataFrame([(row['acct_nbr_ori_1'], val) for _, row in cnt_list.iterrows() for val in row['acct_action_cnt']], columns = ['acct_nbr_ori', 'action_cnt'])
        action_cnt['order'] = action_cnt.groupby('acct_nbr_ori_1').cumcount()+1
        session = pd.DataFrame([(row['acct_nbr_ori_1'], val) for _, row in cnt_list.iterrows() for val in row['session_list']], columns = ['acct_nbr_ori', 'session'])
        session['order'] = session.groupby('acct_nbr_ori_1').cumcount()+1
        action_cnt_session = action_cnt.merge(session, on = ['acct_nbr_ori_1', 'order'])
        action_cnt_session = action_cnt_session.drop(['order'], axis = 1)
        action_cnt_session['order'] = action_cnt_session.groupby('acct_nbr_ori_1').cumcount() + 1
        data = data.merge(action_cnt_session, on=['acct_nbr_ori_1', 'order'], how='left')
        data = data.sort_values(by=['acct_nbr_ori_1', 'order']).reset_index(drop = True)
        data['session'] = data.groupby('acct_nbr_ori_1')['session'].rank(method = 'dense')
```

遇到1時，接續session給值

遇到0時，如果是該客戶第一筆，接續session給值

遇到0時，如果非該客戶第一筆，session+1

(x.acct_nbr_ori, x.drcr)

帳號	轉入/出
00123456789123	1
00123456789123	1
00123456789123	1
00123456789123	2

(cnt_list)

0	1	
00123456789123	0	←啟始值0
00123456789123	1	
00123456789123	1	
00123456789123	0	←判斷方向不同給0

(cnt_list)

insert_list

帳號	轉入/出	acc_cnt	session	
00123456789123	1	0	1	
00123456789123	1	1	1	
00123456789123	1	2	1	
00123456789123	2	0	2	←判斷0時+1

特徵工程函式-1

03.session → 04.flow

連續轉出或轉入總金額

```
def session_total_amt_func(data, mode = 'train', source = 'normal'):
    """
    功能: 捕捉session總金額
    input: 交易資料Dataframe, mode: 訓練或預測模式, 資料源: 警示戶或正常戶
    output: 交易資料Dataframe
    """
    print('session_total_amt_func')
    if mode == 'train' and source == 'normal':
        session_total_amt = data.groupby(['acct_nbr_ori_1', 'session'])['tx_amt'].sum().reset_index().rename(columns = {'tx_amt': 'session_total_amt'})
        session_total_amt['session_total_amt'] = session_total_amt['session_total_amt'].astype(float)
        session_total_amt = session_total_amt[['acct_nbr_ori_1', 'session', 'session_total_amt']]
        data = data.merge(session_total_amt, on = ['acct_nbr_ori_1', 'session'], how = 'left').sort_values(by = ['acct_nbr_ori_1', 'order'])
    else:
        session_total_amt = data.groupby(['acct_nbr_ori', 'session'])['tx_amt'].sum().reset_index().rename(columns = {'tx_amt': 'session_total_amt'})
        session_total_amt['session_total_amt'] = session_total_amt['session_total_amt'].astype(float)
        session_total_amt = session_total_amt[['acct_nbr_ori', 'session', 'session_total_amt']]
        data = data.merge(session_total_amt, on = ['acct_nbr_ori', 'session'], how = 'left').sort_values(by = ['acct_nbr_ori', 'order'])
    return data
```

針對每個帳號、session計算該session總金額

特徵工程函式-1

03.session → 04.flow

連續轉出或轉入該筆當下累計金額

```
def session_accumulated_amt_func(data: F.DataFrame, mode = 'train', source = 'normal'):
    """
    功能: 捕捉session累計金額
    input: 交易資料Dataframe, mode: 訓練或預測模式, 資料源: 警示戶或正常戶
    output: 交易資料Dataframe
    """
    print('session_accumulated_amt_func')
    if mode == 'train' and source == 'normal':
        data = data.sort_values(by=['acct_nbr_ori_1','session','order'])
        accum_txn = data.groupby(['acct_nbr_ori_1','session','order'])['tx_amt'].apply(lambda x: pd.Series(x.cumsum())).reset_index().rename(columns = {'tx_amt': 'session_accumulated_amt'})
        session_accumulated_amt = pd.concat([data[['acct_nbr_ori_1','session','order']], accum_txn[['session_accumulated_amt']]], axis = 1)
        data = data.merge(session_accumulated_amt, on = ['acct_nbr_ori_1', 'session', 'order'], how='left')
    else:
        data = data.sort_values(by=['acct_nbr_ori','session','order'])
        accum_txn = data.groupby(['acct_nbr_ori','session','order'])['tx_amt'].apply(lambda x: pd.Series(x.cumsum())).reset_index().rename(columns = {'tx_amt': 'session_accumulated_amt'})
        session_accumulated_amt = pd.concat([data[['acct_nbr_ori','session','order']], accum_txn[['session_accumulated_amt']]], axis = 1)
        data = data.merge(session_accumulated_amt, on = ['acct_nbr_ori', 'session', 'order'], how='left')
    return data
```

針對每個帳號、session計算該交易金額在此session累積了多少

特徵工程函式-1

03.session → 04.flow

```python
def flow_func(data, mode = 'train', source = 'normal', broadcastVar = 0):
    """
    功能: 捕捉先轉入後轉出行為，給予flow編號
    input: 交易資料Dataframe, mode: 訓練或預測模式, 資料源: 警示戶或正常戶, broadcastVar: flow的default值0
    output: 交易資料Dataframe
    """
    print('flow_func')
    def count_idx(x, broadcastVar):
        #global broadcastVar
        broadcastVar+=1
        flow_list=[]
        for idx in range(0, len(x)):
            if idx != (len(x)-1):  # 不等於最後一個 index
                if (x[idx]-x[idx+1])<0: # 轉出變成轉入
                    flow_list.append(broadcastVar)
                    broadcastVar+=1  # 換flow
                else:
                    flow_list.append(broadcastVar)
            else: # 如果是最後一筆交易
                if (x[idx-1]-x[idx])<0: # 最後一筆是轉入，但前一筆是轉出則判斷是新的flow
                    #broadcastVar+=1
                    flow_list.append(broadcastVar)
                else: # 拿最後一個組合的 broadcastVar 去補結果
                    flow_list.append(broadcastVar)
        return flow_list
    if mode == 'train' and source == 'normal':
        flow_data = data.groupby(['acct_nbr_ori_1'])['drcr'].apply(list).reset_index()
        flow_data['flow'] = flow_data['drcr'].apply(lambda x: count_idx(x, broadcastVar = 0))
        flow_data = flow_data.apply(lambda x: pd.Series(x['flow']), axis = 1).stack().reset_index(level = 1, drop = True).to_frame('flow').join(flow_
        flow_data = flow_data.rename(columns = {'acct_nbr_ori_1': 'acct_nbr_ori_2'}).reset_index(drop = True)
        flow_data['order_2'] = flow_data.groupby('acct_nbr_ori_2').cumcount()+1
        data = pd.merge(data, flow_data, left_on = ['acct_nbr_ori_1', 'order'], right_on = ['acct_nbr_ori_2', 'order_2'], how = 'left')
        data = data.sort_values(['acct_nbr_ori_1', 'order']).reset_index(drop = True)
        data = data.drop(['acct_nbr_ori_2', 'order_2'], axis = 1)

    else:
        flow_data = data.groupby(['acct_nbr_ori'])['drcr'].apply(list).reset_index()
        flow_data['flow'] = flow_data['drcr'].apply(lambda x: count_idx(x, broadcastVar = 0))
        flow_data = flow_data.apply(lambda x: pd.Series(x['flow']), axis = 1).stack().reset_index(level = 1, drop = True).to_frame('flow').join(flow_data[['acct_nbr_ori']], how = 'left')
        flow_data = flow_data.rename(columns = {'acct_nbr_ori': 'acct_nbr_ori_2'}).reset_index(drop = True)
        flow_data['order_2'] = flow_data.groupby('acct_nbr_ori_2').cumcount()+1
        data = pd.merge(data, flow_data, left_on = ['acct_nbr_ori', 'order'], right_on = ['acct_nbr_ori_2', 'order_2'], how = 'left')
        data = data.sort_values(['acct_nbr_ori', 'order']).reset_index(drop = True)
        data = data.drop(['acct_nbr_ori_2', 'order_2'], axis = 1)
    return data
```

(x.acct_nbr_ori, x.drcr)

帳號	轉入/出
00123456789123	2
00123456789123	2
00123456789123	1
00123456789123	2

←判斷轉出變轉入時+1

(flow_list)

帳號	轉入/出	flow
00123456789123	2	1
00123456789123	2	1
00123456789123	1	1
00123456789123	2	2

特徵工程函式-1

03.session → 04.flow

先轉入後轉出Flow之總轉入金額

```
def flow_ttl_amt_2_func(data, mode = 'train', source = 'normal'):
    """
    功能: 計算flow總轉入金額
    input: 交易資料Dataframe, mode: 訓練或預測模式, 資料源: 警示戶或正常戶
    output: 交易資料Dataframe
    """
    print('flow_ttl_amt_2_func')
    if mode == 'train' and source == 'normal':
        flow_ttl_amt_2 = data[data['drcr'] == 2].groupby(['acct_nbr_ori', 'flow'])['tx_amt'].sum().reset_index().rename(columns = {'tx_amt': 'flow_total_amt_2'})
        data = pd.merge(data, flow_ttl_amt_2[['acct_nbr_ori', 'flow', 'flow_total_amt_2']], on = ['acct_nbr_ori', 'flow'], how = 'left')
        data = data.sort_values(['acct_nbr_ori', 'flow', 'order'])
        data = data.fillna({'flow_total_amt_2': 0})

    else:
        flow_ttl_amt_2 = data[data['drcr'] == 2].groupby(['acct_nbr_ori', 'flow'])['tx_amt'].sum().reset_index().rename(columns = {'tx_amt': 'flow_total_amt_2'})
        data = pd.merge(data, flow_ttl_amt_2[['acct_nbr_ori', 'flow', 'flow_total_amt_2']], on = ['acct_nbr_ori', 'flow'], how = 'left')
        data = data.sort_values(['acct_nbr_ori', 'flow', 'order'])
        data = data.fillna({'flow_total_amt_2': 0})
    return data
```

特徵工程函式-1

03.session → 04.flow

- 先轉入後轉出Flow之總轉入/出金額差異占比

```python
def flow_ttl_amt_1_func(data, mode = 'train', source = 'normal'):
    """
    功能: flow總轉出與總轉入金額差異比率
    input: 交易資料Dataframe, mode: 訓練或預測模式, 資料源: 警示戶或正常戶
    output: 交易資料Dataframe
    """
    print('flow_ttl_amt_1_func')
    if mode == 'train' and source == 'normal':
        flow_ttl_amt_1 = data[data['drcr']==1].groupby(['acct_nbr_ori_1', 'flow']).agg({'tx_amt': 'sum'}).rename(columns = {'tx_amt': 'flow_ttl_amt_1'}).reset_index()
        data = pd.merge(data, flow_ttl_amt_1, on = ['acct_nbr_ori_1', 'flow'], how = 'left')
        data['flow_ttl_amt_1'] = data['flow_ttl_amt_1'].fillna(0)
        data['flow_ttl_amt_drcr_ratio'] = abs((data['flow_total_amt_2'] - data['flow_ttl_amt_1'])/(data['flow_total_amt_2']/2 + data['flow_ttl_amt_1']/2)).astype(float)
        data['flow_ttl_amt_drcr_ratio'] = data['flow_ttl_amt_drcr_ratio'].fillna(0)
    else:
        flow_ttl_amt_1 = data[data['drcr']==1].groupby(['acct_nbr_ori', 'flow']).agg({'tx_amt': 'sum'}).rename(columns = {'tx_amt': 'flow_ttl_amt_1'}).reset_index()
        data = pd.merge(data, flow_ttl_amt_1, on = ['acct_nbr_ori', 'flow'], how = 'left')
        data['flow_ttl_amt_1'] = data['flow_ttl_amt_1'].fillna(0)
        data['flow_ttl_amt_drcr_ratio'] = abs((data['flow_total_amt_2'] - data['flow_ttl_amt_1'])/(data['flow_total_amt_2']/2 + data['flow_ttl_amt_1']/2)).astype(float)
        data['flow_ttl_amt_drcr_ratio'] = data['flow_ttl_amt_drcr_ratio'].fillna(0)
    return data
```

觀察此次flow轉入是否都轉出，總轉入/出差異越小，此flow越可疑

特徵工程函式-1

04.flow → 05.約轉設定資訊

計算約轉帳戶數、透過多少約轉通路設定約轉帳號

```
def CFPEBTRFIN_func(data, cfpebtrfin):
    """
    功能: 網銀約轉設定
    input: 交易資料Spark Dataframe, spark
    output: 交易資料Spark Dataframe
    """
    print('CFPEBTRFIN_func')
    data_distinct = data['acct_nbr_ori'].unique()
    data_distinct = pd.DataFrame(data_distinct, columns = ['acct_nbr_ori'])
    cfpebtrfin = cfpebtrfin[['ACNO_OUT','BANK_NO','ACNO_IN','USER_ID_LEVEL','ORI_REQ_DATE','ORI_REQ_BRH','LST_CHG_BRH','LST_MTN_DATE']]
    #轉出帳號(PER)(ACCT)
    cfpebtrfin['ACNO_OUT'] = cfpebtrfin['ACNO_OUT'].str[2:20]
    cfpebtrfin = cfpebtrfin.merge(data_distinct, left_on = 'ACNO_OUT', right_on = 'acct_nbr_ori', how='inner').drop('acct_nbr_ori', axis = 1)
    cfpebtrfin['ACNO_IN'] = cfpebtrfin['ACNO_IN'].str[0:20]
    data_date = data[['acct_nbr_ori', 'tx_date']].drop_duplicates()
    cfpebtrfin = cfpebtrfin.merge(data_date, left_on = 'ACNO_OUT', right_on = 'acct_nbr_ori', how='inner').drop('acct_nbr_ori', axis = 1)
    cfpebtrfin = cfpebtrfin[cfpebtrfin['ORI_REQ_DATE']<=cfpebtrfin['tx_date']]
    acct_acno_in_num = cfpebtrfin.groupby(['ACNO_OUT', 'tx_date']).agg({'ACNO_IN': 'unique'}).rename(columns = {'ACNO_IN': 'acct_acno_in_num'}).reset_index()
    acct_bank_no = cfpebtrfin.groupby(['ACNO_OUT', 'tx_date']).agg({'BANK_NO': 'unique'}).rename(columns = {'BANK_NO': 'acct_bank_no'}).reset_index()
    user_id_level = cfpebtrfin.groupby(['ACNO_OUT', 'tx_date']).agg({'USER_ID_LEVEL': 'unique'}).rename(columns = {'USER_ID_LEVEL': 'user_id_level'}).reset_index()
    req_lst_day = cfpebtrfin.groupby(['ACNO_OUT', 'tx_date']).agg({'ORI_REQ_DATE': 'max'}).rename(columns = {'ORI_REQ_DATE': 'MAX_REQ_DATE'}).reset_index()
    req_lst_day['req_lst_day'] = (pd.to_datetime(req_lst_day['tx_date']) -pd.to_datetime(req_lst_day['MAX_REQ_DATE'])).dt.days
    req_lst_day = req_lst_day.drop('MAX_REQ_DATE', axis = 1)
    ori_req_brh_num = cfpebtrfin.groupby(['ACNO_OUT', 'tx_date']).agg({'ORI_REQ_BRH': 'unique'}).rename(columns = {'ORI_REQ_BRH': 'ori_req_brh_num'}).reset_index()
    # 計算數量
    acct_acno_in_num['acct_acno_in_num'] = acct_acno_in_num['acct_acno_in_num'].apply(lambda x: len(x))
    acct_bank_no['acct_bank_no'] = acct_bank_no['acct_bank_no'].apply(lambda x: len(x))
    user_id_level['user_id_level'] = user_id_level['user_id_level'].apply(lambda x: len(x))
    ori_req_brh_num['ori_req_brh_num'] = ori_req_brh_num['ori_req_brh_num'].apply(lambda x: len(x))
    # 合併cfpebtrfin_main
    cfpebtrfin_main = acct_acno_in_num.merge(acct_bank_no, on = ['ACNO_OUT','tx_date'])
    cfpebtrfin_main = cfpebtrfin_main.merge(user_id_level, on = ['ACNO_OUT','tx_date'])
    cfpebtrfin_main = cfpebtrfin_main.merge(req_lst_day, on = ['ACNO_OUT','tx_date'])
    cfpebtrfin_main = cfpebtrfin_main.merge(ori_req_brh_num, on = ['ACNO_OUT','tx_date'])
    cfpebtrfin_main = cfpebtrfin_main.rename(columns = {'ACNO_OUT': 'acct_nbr_ori'})
    # 合併data
    data = data.merge(cfpebtrfin_main, on=['acct_nbr_ori','tx_date'], how='left')
    data['acct_acno_in_num'] = data['acct_acno_in_num'].fillna(0)
    data['acct_bank_no'] = data['acct_bank_no'].fillna(0)
    data['user_id_level'] = data['user_id_level'].fillna(0)
```

找交易日前設定的約轉帳號

算計約轉帳號數

算計約轉分行數

算計約轉設定通路數 (user_id_level)

算計最後約轉距今天數

網行銀軌跡特徵工程(流程供參)

程式碼

流程僅供參考

Job_SQL語法

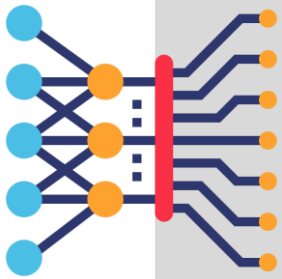
POLICE_DASB_job_cust_pred.sql
POLICE_DASB_salary_acct.sql
POLICE_DASB_job_tracking_log_pred.sql

Job 排程腳本

01.data_cleaning.py
02.feature_engineering_1.py
03.feature_engineering_2.py

Utils函式

config_tbl.py
data_cleansing.py
feature_engineering_2023.py
model_testing.py



Model模型

xgb_model_2022-08-31_fin_v1
second_xgb_model_2022-08-31_fin_v1

特徵工程2

讀取參數檔

03.feature_engineering_2.py

```
##################
####讀取參數檔####
##################
# 建立參數                                          # TODO_5: 請修改config_tbl.py中VIEW及表名
HIVE_VIEW = config.HIVE_VIEW
# 行銀資料表
MB_LOG_VIEW = config.MB_LOG_VIEW                                  # 行銀資料VIEW
MB_ACCESS_LOG = config.MB_ACCESS_LOG                              # 行銀資料檔名
B2C_LOG_VIEW = config.B2C_LOG_VIEW                                # 網銀資料VIEW
B2C_ACCESS_LOG = config.B2C_ACCESS_LOG                            # 網銀資料檔名
print(f'HIVE_VIEW: {HIVE_VIEW}')
print(f'ORACLE_VIEW: {ORACLE_VIEW}')
```

特徵工程2

建帳號、ID、最早最晚交易日表

03.feature_engineering_2.py

```
#########################
###建立帳號、cust_id資料###
#########################
nonwarning = pd.read_csv(f'nonwarning_feature_engineering.csv')
#建立最晚交易日表
nonwarning_1 = nonwarning
nonwarning_cust = nonwarning_1[['acct_nbr_ori', 'cust_id']].drop_duplicates()
res_max = nonwarning_1.groupby('cust_id')['tx_date'].max().reset_index()
res_max = res_max.rename(columns = {'tx_date': 'max_date'})
df_max = res_max[['cust_id', 'max_date']]
#建立最早交易日表
res_min = nonwarning_1.groupby('cust_id')['tx_date'].min().reset_index()
res_min = res_min.rename(columns = {'tx_date': 'min_date'})
df_min = res_min[['cust_id', 'min_date']]
#合併最早交易日表
nonwarning_cust = nonwarning_cust.merge(df_max, on = 'cust_id', how = 'left')
nonwarning_cust = nonwarning_cust.merge(df_min, on = 'cust_id', how = 'left')

#轉換為時間戳記格式
#nonwarning_cust['max_date'] = pd.to_datetime(nonwarning_cust['max_date'], format='%Y-%m-%d %H:%M:%S')
#nonwarning_cust['min_date'] = pd.to_datetime(nonwarning_cust['min_date'], format='%Y-%m-%d %H:%M:%S')
nonwarning_cust['max_date'] = nonwarning_cust['max_date'].apply(lambda x: datetime.strptime(x,'%Y-%m-%d %H:%M:%S'))
nonwarning_cust['min_date'] = nonwarning_cust['min_date'].apply(lambda x: datetime.strptime(x,'%Y-%m-%d %H:%M:%S'))

########################
## 客戶資料 to impala ##
########################
# invalidate_metadata
impala_sql('', impala_logger, mode = 'invalidate_metadata', table_name = f'{HIVE_VIEW}.job_cust_id_pred')
```

acct_nbr_ori	cust_id	min_date	max_date
007371*	A12345*	2023-05-01	2023-05-30
007372*	B12346*	2023-05-02	2023-06-01
007373*	C12347*	2023-05-02	2023-06-01
007374*	D12348*	2023-05-04	2023-06-03

特徵工程前置作業

計算餘額查詢次數

Job_sql: POLICE_DASB_job_tracking_log_pred.sql

```
SELECT company_uid, access_date, SUM(CASE WHEN category = 'MB_limit' THEN times END) MB_limit, SUM(CASE WHEN category = 'MB_check' THEN times END
FROM(
    SELECT company_uid, access_date, category, COUNT(DISTINCT login_log_key) as times
    FROM (
        SELECT
            company_uid,
            login_log_key,
            to_date(access_time) AS access_date,
            error_desc,
            menu_id,
            CASE
                WHEN menu_id IN ('MDS01', 'MDS0101', 'MDS04', 'MDS0401', 'MDS0402') THEN 'MB_check' --查餘額
                WHEN menu_id IN ('MPS1302') THEN 'MB_limit' --行銀調額
            END AS category
        FROM ODS_T_VIEW.MB_ACCESS_LOG
        WHERE company_uid IN (
            SELECT distinct cust_id
            FROM usr_julian_liu.job_cust_id_pred
        )
        AND menu_id in ('MDS01', 'MDS0101', 'MDS04', 'MDS0401', 'MDS0402','MPS1302')
        AND from_unixtime(unix_timestamp(access_date), 'yyyy-MM-dd') BETWEEN (SELECT min(min_date) FROM usr_julian_liu.job_cust_id_pred)
        AND (SELECT max(max_date) min_date FROM usr_julian_liu.job_cust_id_pred)
        UNION ALL
        SELECT
            company_uid,
            login_log_key,
            to_date(access_time) AS access_date,
            error_desc,
            menu_id,
            CASE
                WHEN menu_id in ('CDS0401', 'CDS04', 'CDS01', 'CDS0102', 'CBO03', 'CDF02', 'CDF04') THEN 'EB_check' --查餘額
            END as category
        FROM ODS_T_VIEW.B2C_ACCESS_LOG
        WHERE company_uid IN (
            SELECT distinct cust_id
            FROM usr_julian_liu.job_cust_id_pred
        )
        AND menu_id in ('CDS0401', 'CDS04', 'CDS01', 'CDS0102', 'CBO03', 'CDF02', 'CDF04')
        AND from_unixtime(unix_timestamp(access_date), 'yyyy-MM-dd') BETWEEN (SELECT min(min_date) FROM usr_julian_liu.job_cust_id_pred)
        AND (SELECT max(max_date) min_date FROM usr_julian_liu.job_cust_id_pred)
    ) t
    GROUP BY company_uid, access_date, category
) p
GROUP BY company_uid, access_date
```

限縮軌跡資料撈取範圍

ID	access_date	category
A12345678*	2023-05-01	MB_check
A12345678*	2023-05-02	MB_check
A12345678*	2023-05-02	MB_limit
A12345678*	2023-05-04	MB_check

ID	access_date	MB_check	MB_limit
A12345678*	2023-05-01	1	0
A12345678*	2023-05-02	1	1
A12345678*	2023-05-04	1	0

特徵工程2

透過腳本檔建立資料表

03.feature_engineering_2.py

```
###################################
#### 由impala計算網行銀查詢次數 ####
###################################
try:
    #建立IP變更次數資料
    tbl_name = 'job_tracking_log_pred'
    # Create impala table
    impala_sql(f'POLICE_DASB_{tbl_name}_tmp_file.sql', impala_logger, mode='overwrite', table_name = f'{HIVE_VIEW}.POLICE_DASB_{tbl_name}')
    print('Tracking_log_alarm Table Creation done!')
except Exception:
    logging.exception('message')
    print('Tracking_log_alarm Creation failed.')
```

透過impala建立資料表

特徵工程2

合併網行銀軌跡資料表

03.feature_engineering_2.py

```
##############################
#### 由Hadoop撈取查餘額次數 ##
##############################
try:
    print('Merging Table started...')
    # Hadoop撈取查餘額次數
    tbl_name = 'job_tracking_log_pred'
    # Create Hive table
    query = f'''
    SELECT *
    FROM usr_julian_liu.POLICE_DASB_{tbl_name}
    '''
    # impala to Hive
    sdf_tracking_log = spark.sql(query)
    sdf_tracking_log.show(2)
    df_tracking_log= sdf_tracking_log.toPandas()
    print(type(df_tracking_log['access_date']))
    df_tracking_log['access_date'] = pd.to_datetime(df_tracking_log['access_date'])
    nonwarning['tx_date'] = pd.to_datetime(nonwarning['tx_date'])
    nonwarning = nonwarning.merge(df_tracking_log, left_on = ['cust_id', 'tx_date'], right_on = ['company_uid', 'access_date'], how = 'left')
    nonwarning['mb_limit'] = nonwarning['mb_limit'].fillna(0)
    nonwarning['mb_check'] = nonwarning['mb_check'].fillna(0)
    nonwarning['eb_check'] = nonwarning['eb_check'].fillna(0)
    nonwarning = nonwarning.drop('access_date', axis = 1)
    nonwarning = nonwarning.drop('company_uid', axis = 1)
    nonwarning[nonwarning['mb_limit']!= 0].head(2)
    print('Merging Table done!')
except Exception:
    logging.exception('message')
    print('Tracking_log_alarm Creation failed.')
```

以統編、登入日期合併

Q&A


Taipei Fubon Bank
Thank You!