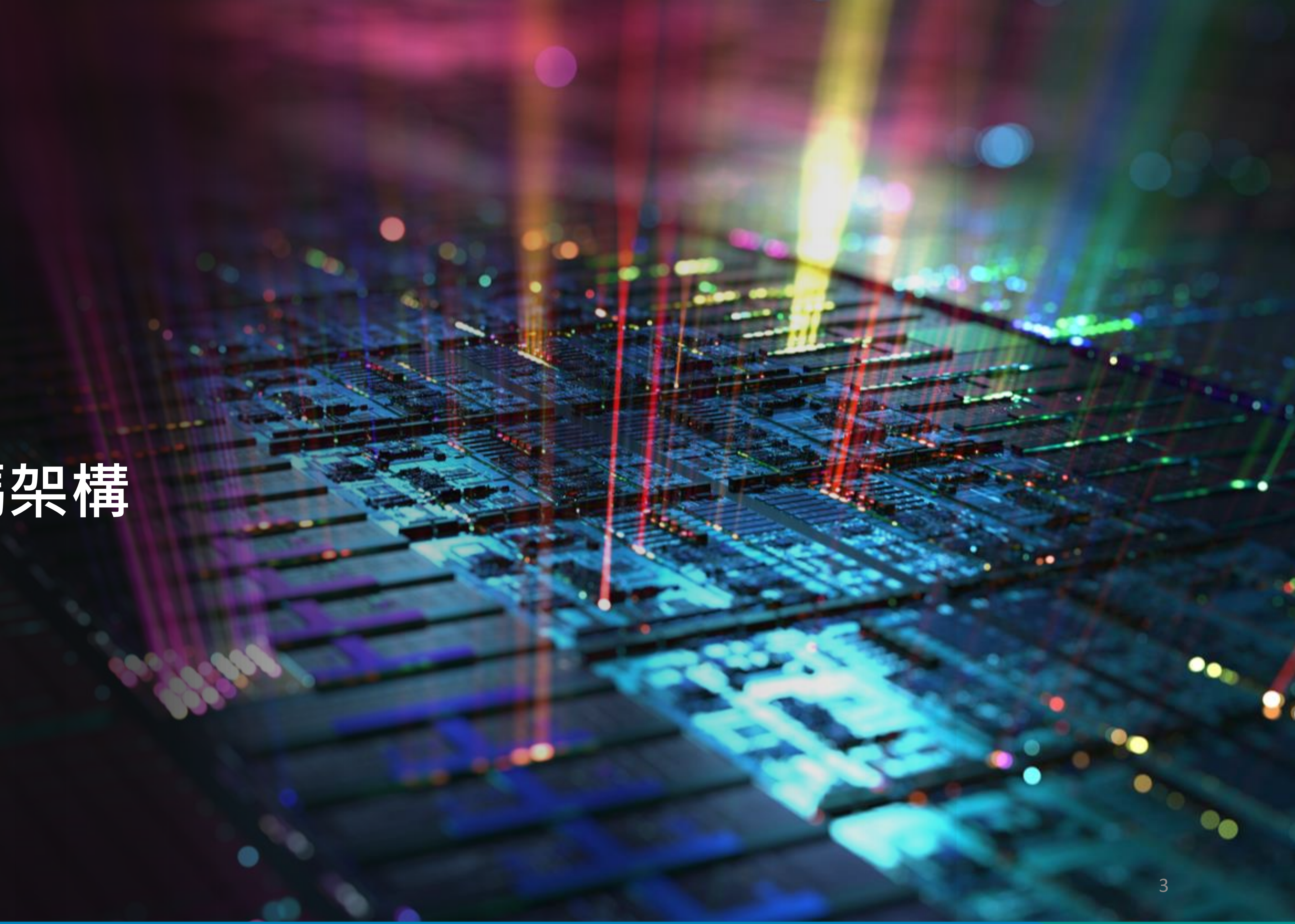
台北富邦銀行 Taipei Fubon Bank

鷹眼識詐聯盟技術交流

數據科學部 劉晉儒 2024.01.04

AGENDA



01.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
04.cust_info.py
05.prediction_moratorium.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

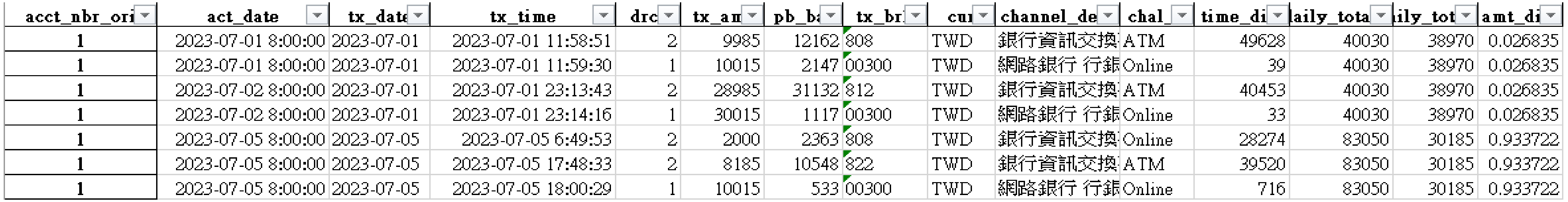
02.交易資料歸戶

資料準備

nonwarning_feature_engineering_{min_date}_{max_date}_v2.csv

No	英文名稱	中文名稱
1	acct_nbr_ori	母帳號
2	acct_nbr	子帳號
3	cust_id	客戶統編
4	act_date	帳務日
5	tx_date	交易日
6	tx_time	交易時間
7	drcr	轉入轉出
8	tx_amt	交易金額(台幣)
9	pb_bal	餘額(台幣)
10	tx_brh	交易分行
11	cur	幣別
12	channel_desc	交易通路
13	chal_1	通路說明
14	time_diff	上一筆交易時間差
15	daily_total_1	當日交易總轉出
16	daily_total_2	當日交易總轉入
17	amt_diff	當日交易總轉入/轉出差額比例
18	tx_brh_cnt	提領分行數
19	income_withdraw_ratio	單筆交易餘額占比

No	英文名稱	中文名稱
20	action_cnt	session連續借貸次數
21	session_total_amt	session總金額
22	session_accumulated_amt	session該筆當下累計金額
23	flow_total_amt_2	flow總轉入金額
24	flow_tx_amt_ratio	flow該筆金額/總轉入金額
25	flow_tx_amt_seq	flow該筆當下累計金額
26	flow_tx_amt_seq_ratio	flow該筆當下累計金額佔flow總金額比例
27	flow_ttl_amt_1	flow總轉出金額
28	flow_ttl_amt_drcr_ratio	flow總轉出/轉入差額比
29	flow_avg_time_diff	flow平均交易時間差
30	active_days	開戶至交易日數
31	acct_acno_in_num	設定約轉帳戶數
32	acct_bank_no	設定約轉行庫數
33	user_id_level	約轉設定通路(1網銀、5新興商務網)
34	req_lst_day	約轉最後建檔日及交易日時間差
35	ori_req_brh_num	約轉建檔分行數
36	EB_check	網銀餘額查詢次數
37	MB_check	行銀餘額查詢次數
38	MB_limit	行銀轉帳額度調整次數



acct_nbr_ori	act_date	tx_date	tx_time	drc	tx_am	pb_ba	tx_br	cur	channel_de	chal_	time_di	laily_tota	ily_tot	amt_di
1	2023-07-01 8:00:00	2023-07-01	2023-07-01 11:58:51	2	9985	12162	808	TWD	銀行資訊交換	ATM	49628	40030	38970	0.026835
1	2023-07-01 8:00:00	2023-07-01	2023-07-01 11:59:30	1	10015	2147	00300	TWD	網路銀行 行銀	Online	39	40030	38970	0.026835
1	2023-07-02 8:00:00	2023-07-01	2023-07-01 23:13:43	2	28985	31132	812	TWD	銀行資訊交換	ATM	40453	40030	38970	0.026835
1	2023-07-02 8:00:00	2023-07-01	2023-07-01 23:14:16	1	30015	1117	00300	TWD	網路銀行 行銀	Online	33	40030	38970	0.026835
1	2023-07-05 8:00:00	2023-07-05	2023-07-05 6:49:53	2	2000	2363	808	TWD	銀行資訊交換	Online	28274	83050	30185	0.933722
1	2023-07-05 8:00:00	2023-07-05	2023-07-05 17:48:33	2	8185	10548	822	TWD	銀行資訊交換	ATM	39520	83050	30185	0.933722
1	2023-07-05 8:00:00	2023-07-05	2023-07-05 18:00:29	1	10015	533	00300	TWD	網路銀行 行銀	Online	716	83050	30185	0.933722

資料準備

01.建立帳號對應表 → 02.平均每日轉出金額 → 03.夜間ATM轉出次數 → 04.ATM交易時間差

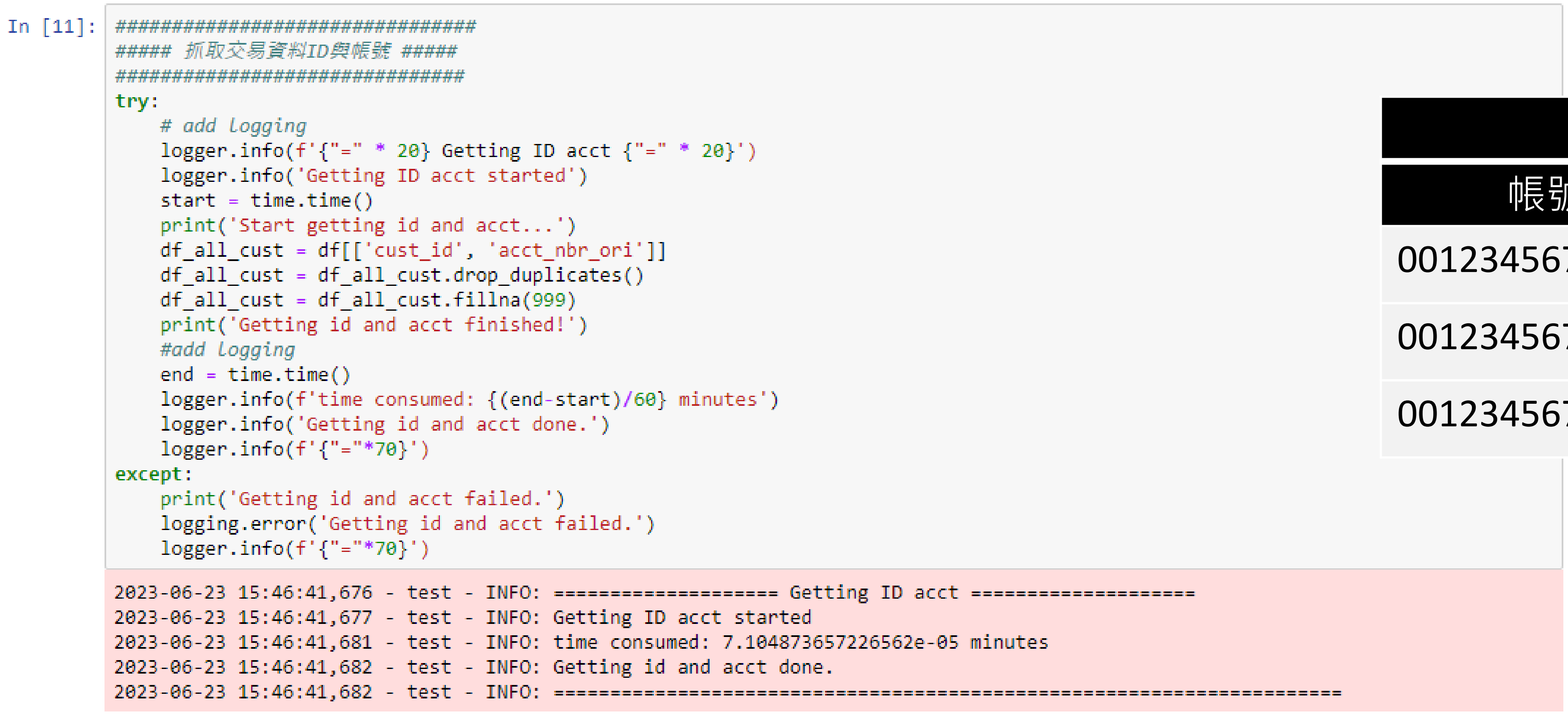
04.cust_info.py

```
#####################
###讀取特徵工程資料###
#####################
nonwarning = pd.read_csv(f'nonwarning_feature_engineering_{min_date}_{max_date}_v2.csv')

################################
### 選取建模所需特定欄位、切xy ###
################################
selected_cols = ['acct_nbr_ori', 'acct_nbr', 'cust_id', 'act_date', 'tx_date', 'tx_time', 'drcr', 'tx_amt', 'pb_bal', 'tx_brh', 'cur',
                 'channel_desc', 'chal_1', 'time_diff', 'daily_total_1', 'daily_total_2', 'amt_diff', 'tx_brh_cnt',
                 'income_withdraw_ratio', 'action_cnt', 'session_total_amt', 'session_accumulated_amt', 'flow_total_amt_2',
                 'flow_tx_amt_ratio', 'flow_tx_amt_seq', 'flow_tx_amt_seq_ratio', 'flow_ttl_amt_1', 'flow_ttl_amt_drcr_ratio',
                 'flow_avg_time_diff', 'active_days', 'acct_acno_in_num', 'acct_bank_no', 'user_id_level', 'req_lst_day',
                 'ori_req_brh_num', 'hour', 'small_bal', 'eb_check', 'mb_check', 'mb_limit']
df = nonwarning[selected_cols]
```

交易資料歸集於帳戶

01.建立帳號對應表 → 02.平均每日轉出金額 → 03.夜間ATM轉出次數 → 04.ATM交易時間差

04.cust_info.py



```
In [11]: ##################################
         ##### 抓取交易資料ID與帳號 #####
         ################################
         try:
             # add logging
             logger.info(f'{"=" * 20} Getting ID acct {"=" * 20}')
             logger.info('Getting ID acct started')
             start = time.time()
             print('Start getting id and acct...')
             df_all_cust = df[['cust_id', 'acct_nbr_ori']]
             df_all_cust = df_all_cust.drop_duplicates()
             df_all_cust = df_all_cust.fillna(999)
             print('Getting id and acct finished!')
             #add logging
             end = time.time()
             logger.info(f'time consumed: {(end-start)/60} minutes')
             logger.info('Getting id and acct done.')
             logger.info(f'{"="*70}')
         except:
             print('Getting id and acct failed.')
             logging.error('Getting id and acct failed.')
             logger.info(f'{"="*70}')
```

```
2023-06-23 15:46:41,676 - test - INFO: ==================== Getting ID acct ====================
2023-06-23 15:46:41,677 - test - INFO: Getting ID acct started
2023-06-23 15:46:41,681 - test - INFO: time consumed: 7.104873657226562e-05 minutes
2023-06-23 15:46:41,682 - test - INFO: Getting id and acct done.
2023-06-23 15:46:41,682 - test - INFO: ======================================================================
```

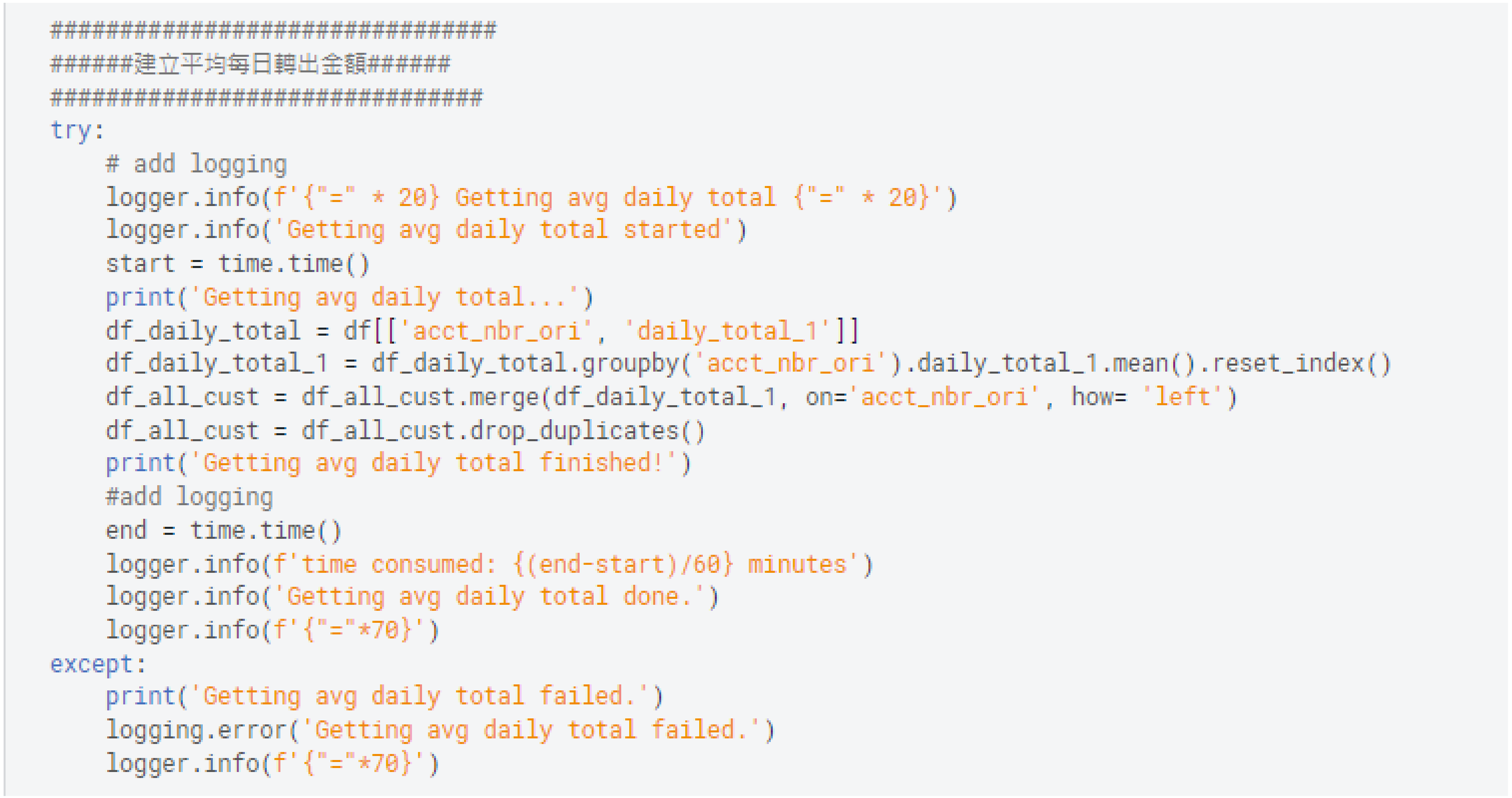
df_all_cust

帳號	ID
00123456789123	A123456789
00123456789124	B123456789
00123456789125	C123456789

交易資料歸集於帳戶

01.建立帳號對應表 → 02.平均每日轉出金額 → 03.夜間ATM轉出次數 → 04.ATM交易時間差

04.cust_info.py

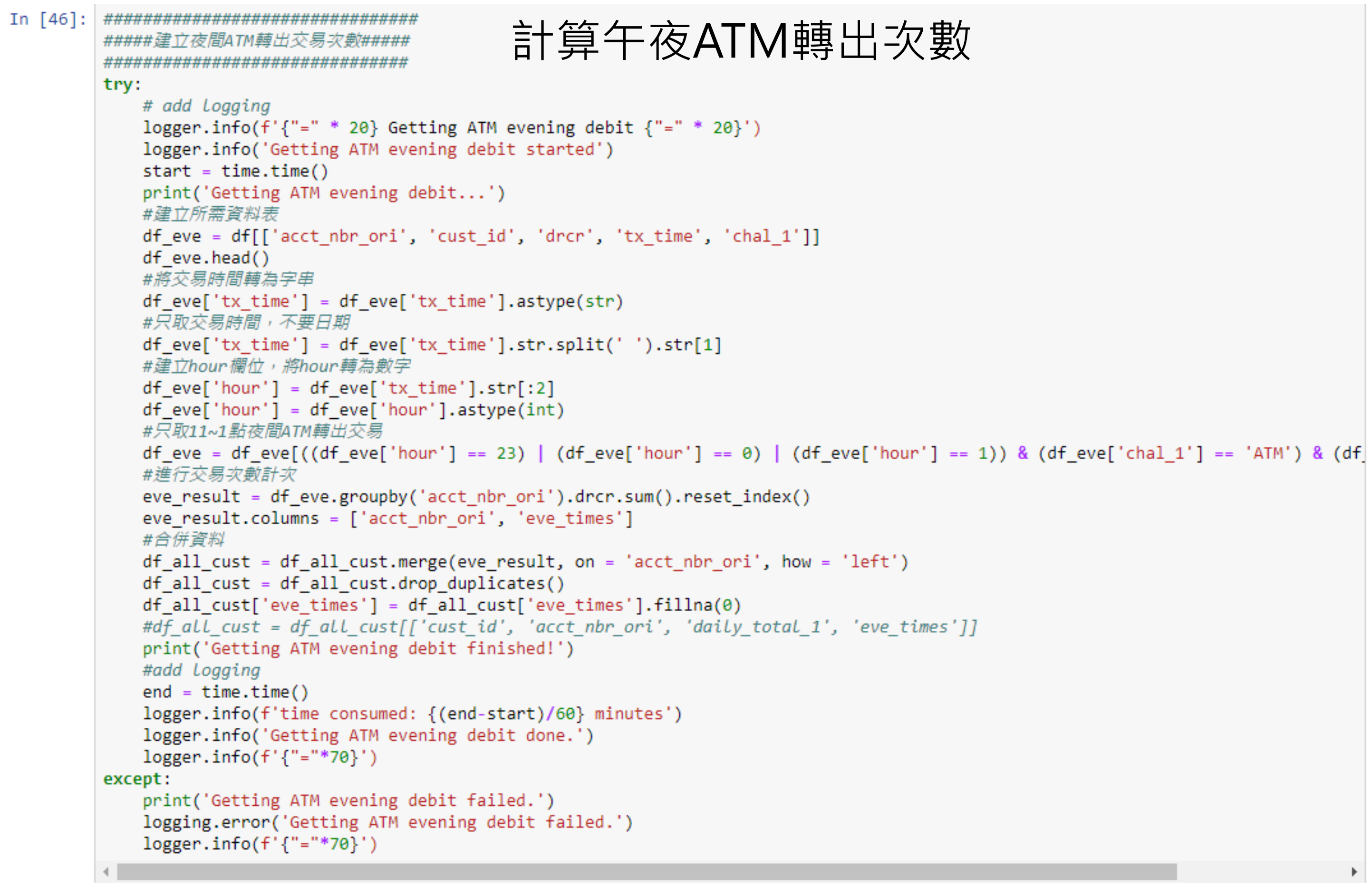


```
################################
######建立平均每日轉出金額######
###############################
try:
    # add logging
    logger.info(f'{"=" * 20} Getting avg daily total {"=" * 20}')
    logger.info('Getting avg daily total started')
    start = time.time()
    print('Getting avg daily total...')
    df_daily_total = df[['acct_nbr_ori', 'daily_total_1']]
    df_daily_total_1 = df_daily_total.groupby('acct_nbr_ori').daily_total_1.mean().reset_index()
    df_all_cust = df_all_cust.merge(df_daily_total_1, on='acct_nbr_ori', how= 'left')
    df_all_cust = df_all_cust.drop_duplicates()
    print('Getting avg daily total finished!')
    #add logging
    end = time.time()
    logger.info(f'time consumed: {(end-start)/60} minutes')
    logger.info('Getting avg daily total done.')
    logger.info(f'{"="*70}')
except:
    print('Getting avg daily total failed.')
    logging.error('Getting avg daily total failed.')
    logger.info(f'{"="*70}')
```

交易資料歸集於帳戶

01.建立帳號對應表 → 02.平均每日轉出金額 → 03.夜間ATM轉出次數 → 04.ATM交易時間差

04.cust_info.py



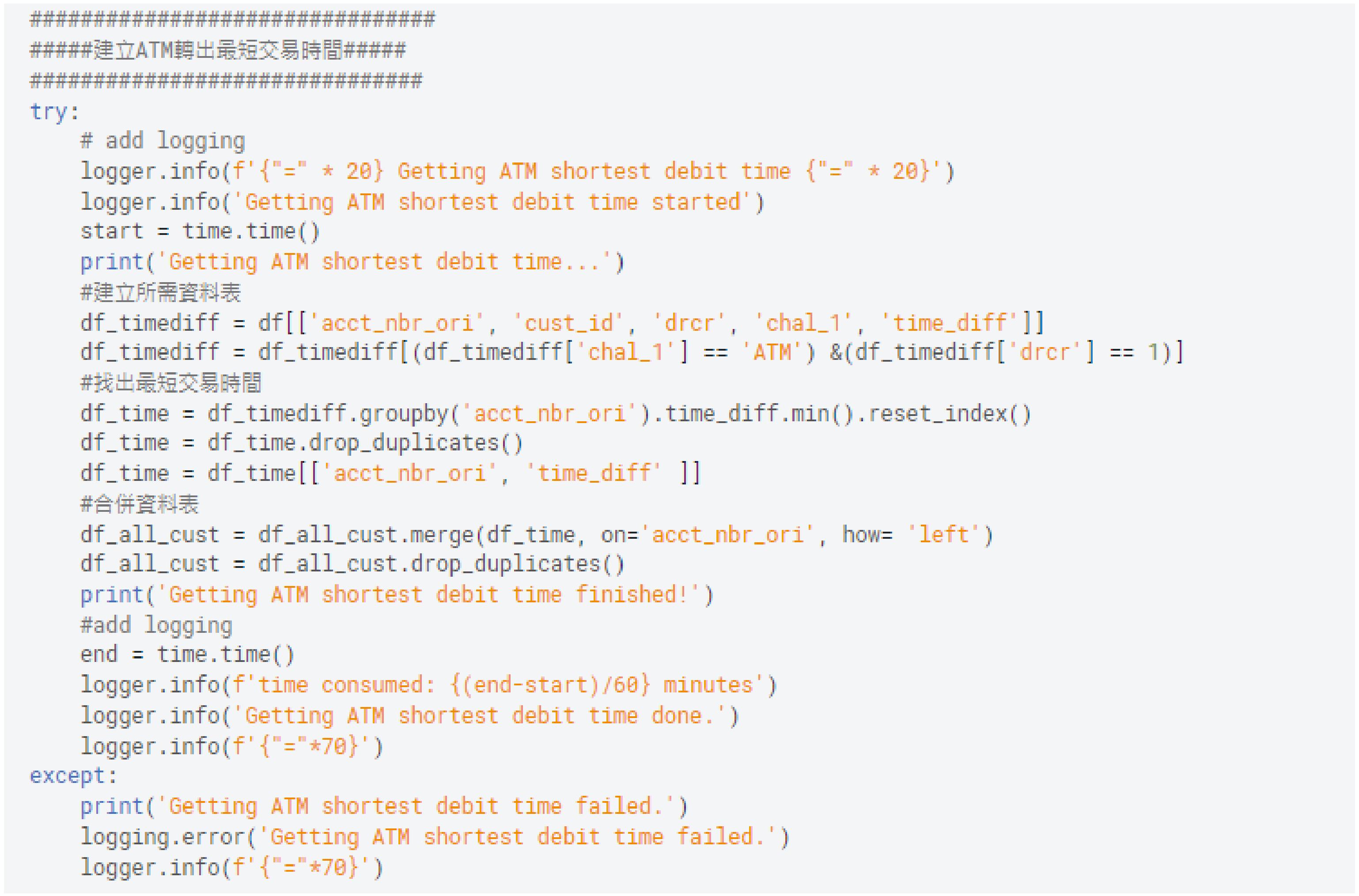
計算午夜ATM轉出次數

```
In [46]: ##################################
         #####建立夜間ATM轉出交易次數#####
         ################################
         try:
             # add logging
             logger.info(f'{"=" * 20} Getting ATM evening debit {"=" * 20}')
             logger.info('Getting ATM evening debit started')
             start = time.time()
             print('Getting ATM evening debit...')
             #建立所需資料表
             df_eve = df[['acct_nbr_ori', 'cust_id', 'drcr', 'tx_time', 'chal_1']]
             df_eve.head()
             #將交易時間轉為字串
             df_eve['tx_time'] = df_eve['tx_time'].astype(str)
             #只取交易時間，不要日期
             df_eve['tx_time'] = df_eve['tx_time'].str.split(' ').str[1]
             #建立hour欄位，將hour轉為數字
             df_eve['hour'] = df_eve['tx_time'].str[:2]
             df_eve['hour'] = df_eve['hour'].astype(int)
             #只取11~1點夜間ATM轉出交易
             df_eve = df_eve[((df_eve['hour'] == 23) | (df_eve['hour'] == 0) | (df_eve['hour'] == 1)) & (df_eve['chal_1'] == 'ATM') & (df_
             #進行交易次數計次
             eve_result = df_eve.groupby('acct_nbr_ori').drcr.sum().reset_index()
             eve_result.columns = ['acct_nbr_ori', 'eve_times']
             #合併資料
             df_all_cust = df_all_cust.merge(eve_result, on = 'acct_nbr_ori', how = 'left')
             df_all_cust = df_all_cust.drop_duplicates()
             df_all_cust['eve_times'] = df_all_cust['eve_times'].fillna(0)
             #df_all_cust = df_all_cust[['cust_id', 'acct_nbr_ori', 'daily_total_1', 'eve_times']]
             print('Getting ATM evening debit finished!')
             #add logging
             end = time.time()
             logger.info(f'time consumed: {(end-start)/60} minutes')
             logger.info('Getting ATM evening debit done.')
             logger.info(f'{"="*70}')
         except:
             print('Getting ATM evening debit failed.')
             logging.error('Getting ATM evening debit failed.')
             logger.info(f'{"="*70}')
```

交易資料歸集於帳戶

01.建立帳號對應表 → 02.平均每日轉出金額 → 03.夜間ATM轉出次數 → 04.ATM交易時間差

04.cust_info.py

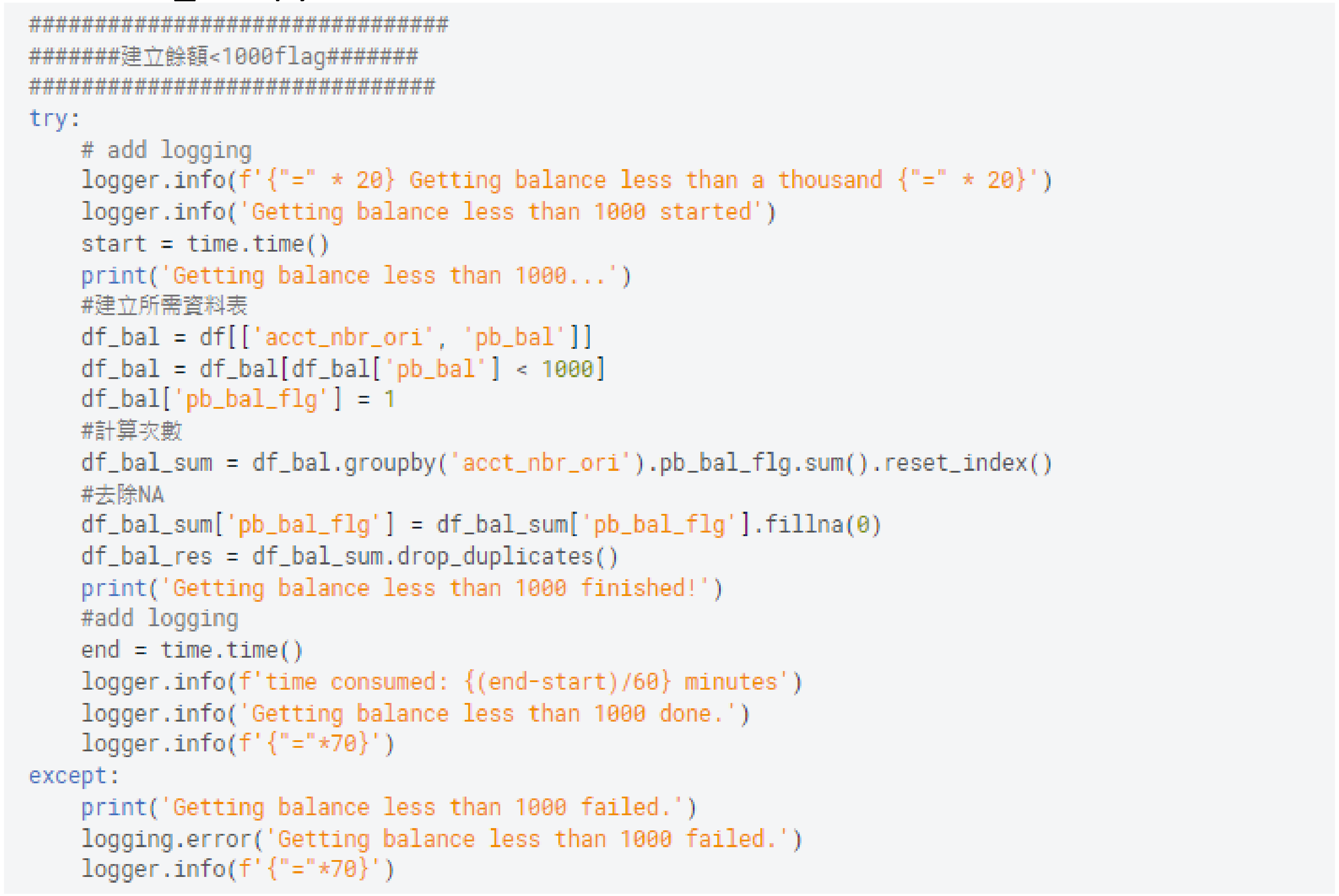


```
################################
#####建立ATM轉出最短交易時間#####
###############################
try:
    # add logging
    logger.info(f'{"=" * 20} Getting ATM shortest debit time {"=" * 20}')
    logger.info('Getting ATM shortest debit time started')
    start = time.time()
    print('Getting ATM shortest debit time...')
    #建立所需資料表
    df_timediff = df[['acct_nbr_ori', 'cust_id', 'drcr', 'chal_1', 'time_diff']]
    df_timediff = df_timediff[(df_timediff['chal_1'] == 'ATM') &(df_timediff['drcr'] == 1)]
    #找出最短交易時間
    df_time = df_timediff.groupby('acct_nbr_ori').time_diff.min().reset_index()
    df_time = df_time.drop_duplicates()
    df_time = df_time[['acct_nbr_ori', 'time_diff' ]]
    #合併資料表
    df_all_cust = df_all_cust.merge(df_time, on='acct_nbr_ori', how= 'left')
    df_all_cust = df_all_cust.drop_duplicates()
    print('Getting ATM shortest debit time finished!')
    #add logging
    end = time.time()
    logger.info(f'time consumed: {(end-start)/60} minutes')
    logger.info('Getting ATM shortest debit time done.')
    logger.info(f'{"="*70}')
except:
    print('Getting ATM shortest debit time failed.')
    logging.error('Getting ATM shortest debit time failed.')
    logger.info(f'{"="*70}')
```

交易資料歸集於帳戶

05.餘額<1000次數 → 06.餘額查詢次數 → 07.資料合併

04.cust_info.py

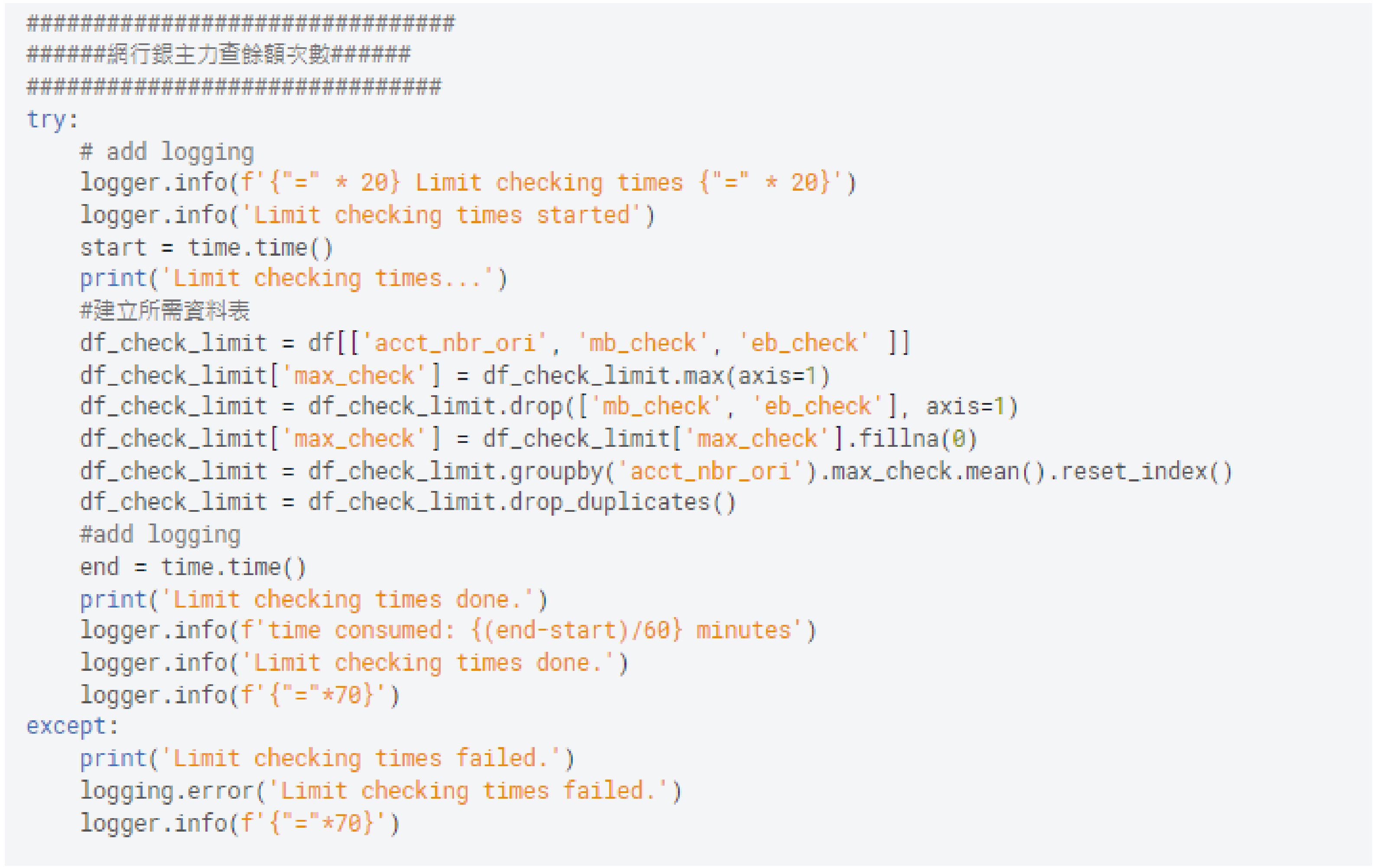


```python
################################
#######建立餘額<1000flag#######
###############################
try:
    # add logging
    logger.info(f'{"=" * 20} Getting balance less than a thousand {"=" * 20}')
    logger.info('Getting balance less than 1000 started')
    start = time.time()
    print('Getting balance less than 1000...')
    #建立所需資料表
    df_bal = df[['acct_nbr_ori', 'pb_bal']]
    df_bal = df_bal[df_bal['pb_bal'] < 1000]
    df_bal['pb_bal_flg'] = 1
    #計算次數
    df_bal_sum = df_bal.groupby('acct_nbr_ori').pb_bal_flg.sum().reset_index()
    #去除NA
    df_bal_sum['pb_bal_flg'] = df_bal_sum['pb_bal_flg'].fillna(0)
    df_bal_res = df_bal_sum.drop_duplicates()
    print('Getting balance less than 1000 finished!')
    #add logging
    end = time.time()
    logger.info(f'time consumed: {(end-start)/60} minutes')
    logger.info('Getting balance less than 1000 done.')
    logger.info(f'{"="*70}')
except:
    print('Getting balance less than 1000 failed.')
    logging.error('Getting balance less than 1000 failed.')
    logger.info(f'{"="*70}')
```

交易資料歸集於帳戶

05.餘額<1000次數 → 06.餘額查詢次數 → 07.資料合併

04.cust_info.py



```
################################
######網行銀主力查餘額次數######
###############################
try:
    # add logging
    logger.info(f'{"=" * 20} Limit checking times {"=" * 20}')
    logger.info('Limit checking times started')
    start = time.time()
    print('Limit checking times...')
    #建立所需資料表
    df_check_limit = df[['acct_nbr_ori', 'mb_check', 'eb_check' ]]
    df_check_limit['max_check'] = df_check_limit.max(axis=1)
    df_check_limit = df_check_limit.drop(['mb_check', 'eb_check'], axis=1)
    df_check_limit['max_check'] = df_check_limit['max_check'].fillna(0)
    df_check_limit = df_check_limit.groupby('acct_nbr_ori').max_check.mean().reset_index()
    df_check_limit = df_check_limit.drop_duplicates()
    #add logging
    end = time.time()
    print('Limit checking times done.')
    logger.info(f'time consumed: {(end-start)/60} minutes')
    logger.info('Limit checking times done.')
    logger.info(f'{"="*70}')
except:
    print('Limit checking times failed.')
    logging.error('Limit checking times failed.')
    logger.info(f'{"="*70}')
```

交易資料歸集於帳戶

05.餘額<1000次數 → 06.餘額查詢次數 → 07.資料合併

04.cust_info.py

```
################################
#####合併網行銀及低於1000餘額#####
###############################
try:
    # add logging
    logger.info(f'{"=" * 20} Data merge {"=" * 20}')
    logger.info('Data merge started')
    start = time.time()
    print('Data merging...')
    #合併AUM、網行銀及低於1000餘額
    df_all_cust = df_all_cust.merge(df_cust, on = 'cust_id', how= 'left')
    df_all_cust = df_all_cust.drop_duplicates()
    df_all_cust = df_all_cust.fillna(0)
    df_all_cust = df_all_cust.merge(df_bal_res, on='acct_nbr_ori', how='left')
    #合併Max limit
    result = df_all_cust.merge(df_check_limit, on='acct_nbr_ori', how='left')
    result = result.drop_duplicates()
    result = fe.acct_nbr_ori_digits(result, 'acct_nbr_ori')
    result = result.fillna(0)
    #add logging
    end = time.time()
    print('Data merge done.')
    logger.info(f'time consumed: {(end-start)/60} minutes')
    logger.info('Data merge done.')
    logger.info(f'{"="*70}')
except:
    print('Data merge failed.')
    logging.error('Data merge failed.')
    logger.info(f'{"="*70}')
```

03.模型預測

程式碼

Job_SQL

POLICE_DASB_job_cust_pred.sql
POLICE_DASB_salary_acct.sql
POLICE_DASB_job_tracking_log_pred.sql

Job 排程

01.data_cleaning.py
02.feature_engineering_1.py
03.feature_engineering_2.py
04.cust_info.py
05.prediction_moratorium.py

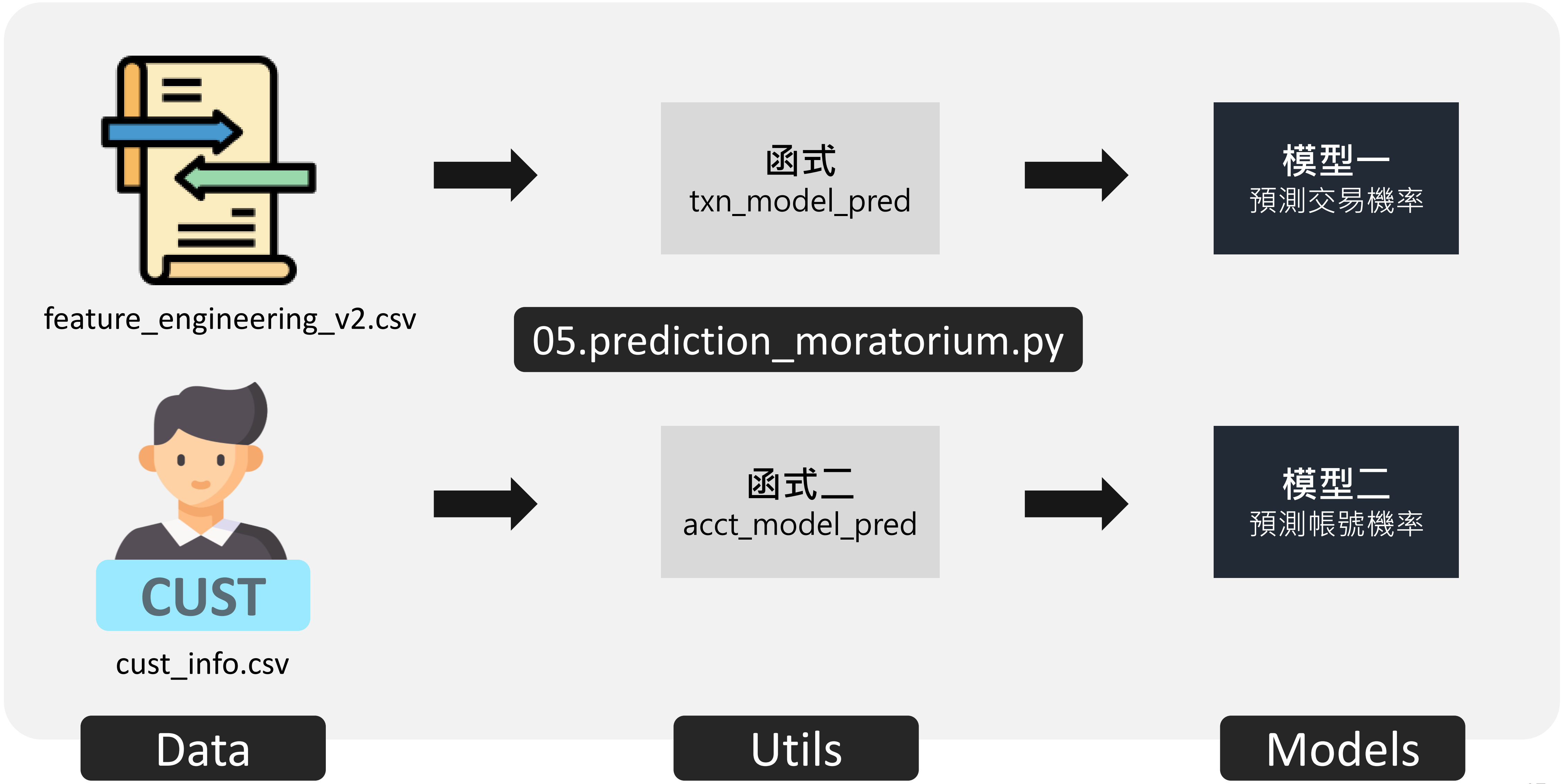
Utils函式

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

預測流程架構


feature_engineering_v2.csv
函式
txn_model_pred
模型一
預測交易機率
05.prediction_moratorium.py
CUST
cust_info.csv
函式二
acct_model_pred
模型二
預測帳號機率
Data
Utils
Models

異常機率計數

模型一
預測交易異常

帳戶 Acct	轉入轉出 drcr	交易金額 txn_amt	餘額 pb_bal	…	連續轉出/入 session	先入後出 flow	交易異常機率
Acc1	1	5000	15,500	…	0	1	0.57
	2	5000	20,500		1	2	0.856
	1	10,000	10,500		2	2	0.95
	1	10,000	500		2	2	0.97

歸戶

模型二
預測帳號異常

帳戶 Acct	0~0.05 x1	0.05~0.01 x2	…	0.5~0.55	0.55~0.6	…	0.85~0.9	0.9~0.95 x19	0.95~1 x20
Acc1	0	0	…	1	0	…	1	0	2

【過程】
1.將模型一之交易異常機率進行裝箱，計算交易異常筆數占比計算
2.補充交易面資訊(ex.夜間ATM轉出交易次數、餘額<1000、每日總轉出金額 、ATM轉出最短時間 、網行銀查詢次數...)

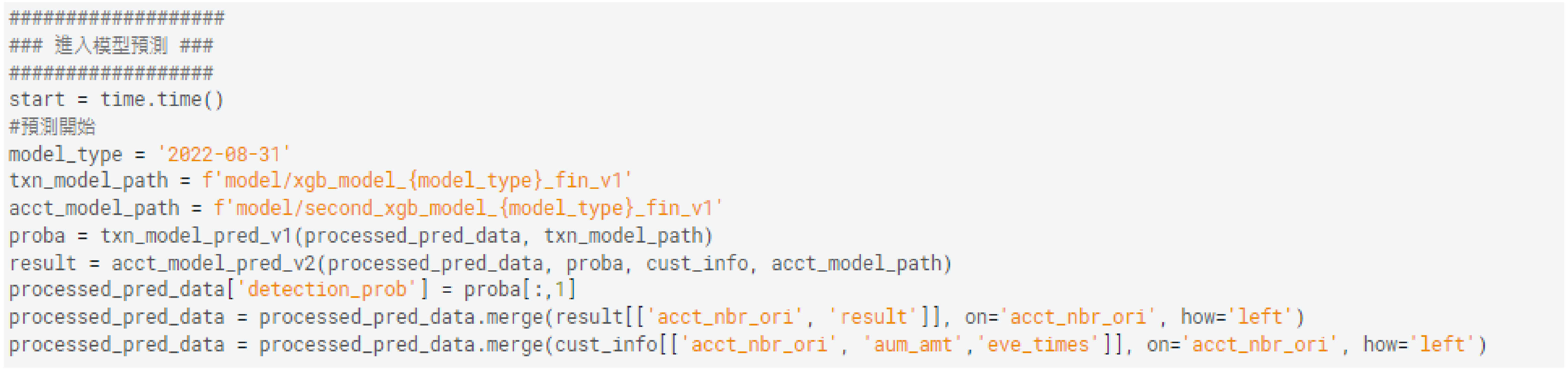
預測模型

XGBoost(序列式生成樹) +

模型一
預測交易機率 → xgb_model_2022-08-31_v1

模型二
預測帳號機率 → second_xgb_model_2022-08-31_v1

05.prediction_moratorium.py

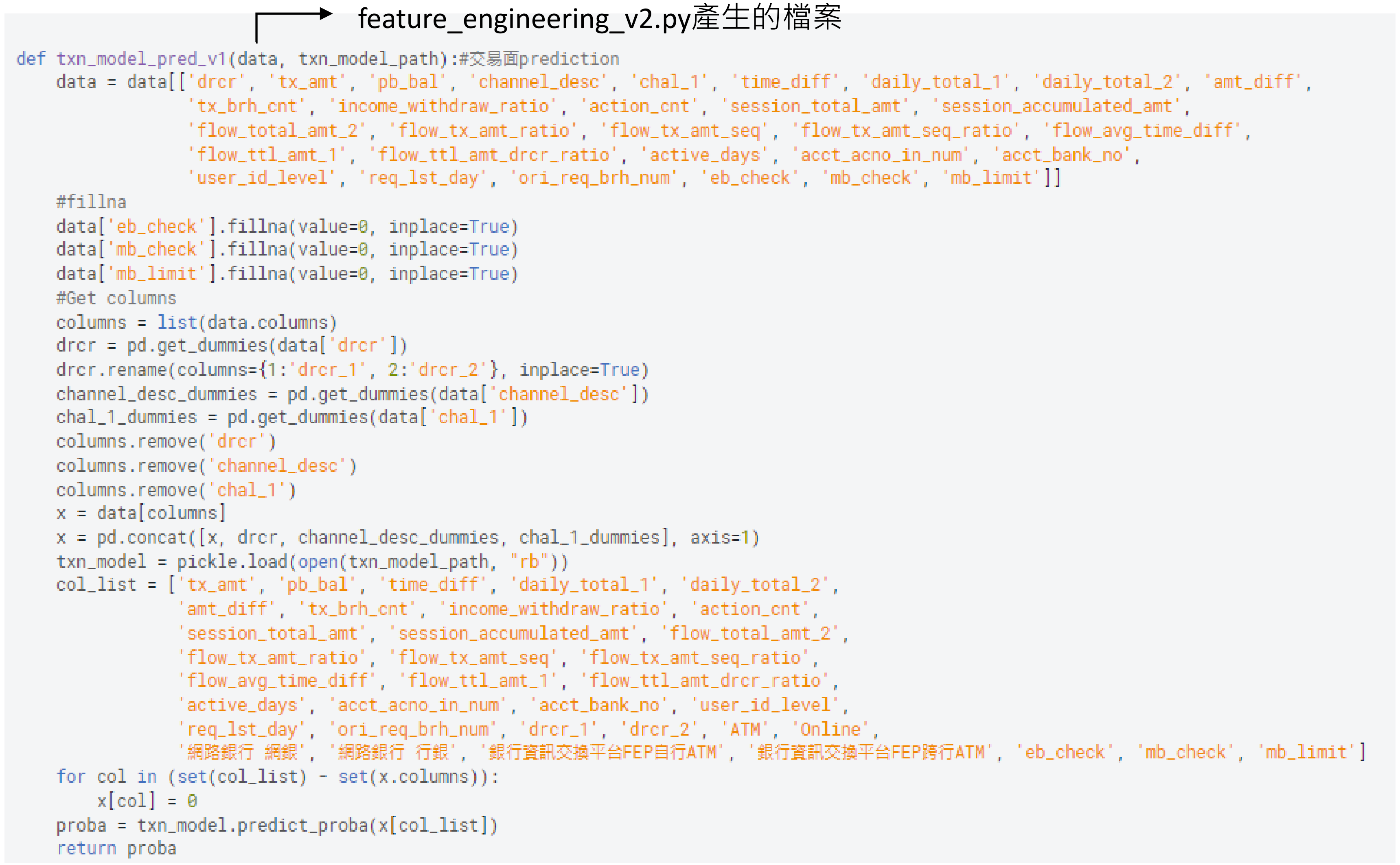


```
###################
### 進入模型預測 ###
##################
start = time.time()
#預測開始
model_type = '2022-08-31'
txn_model_path = f'model/xgb_model_{model_type}_fin_v1'
acct_model_path = f'model/second_xgb_model_{model_type}_fin_v1'
proba = txn_model_pred_v1(processed_pred_data, txn_model_path)
result = acct_model_pred_v2(processed_pred_data, proba, cust_info, acct_model_path)
processed_pred_data['detection_prob'] = proba[:,1]
processed_pred_data = processed_pred_data.merge(result[['acct_nbr_ori', 'result']], on='acct_nbr_ori', how='left')
processed_pred_data = processed_pred_data.merge(cust_info[['acct_nbr_ori', 'aum_amt','eve_times']], on='acct_nbr_ori', how='left')
```

交易資料建立特徵、dummy

model_testing → txn_model_predict

feature_engineering_v2.py產生的檔案



```python
def txn_model_pred_v1(data, txn_model_path):#交易面prediction
    data = data[['drcr', 'tx_amt', 'pb_bal', 'channel_desc', 'chal_1', 'time_diff', 'daily_total_1', 'daily_total_2', 'amt_diff',
                 'tx_brh_cnt', 'income_withdraw_ratio', 'action_cnt', 'session_total_amt', 'session_accumulated_amt',
                 'flow_total_amt_2', 'flow_tx_amt_ratio', 'flow_tx_amt_seq', 'flow_tx_amt_seq_ratio', 'flow_avg_time_diff',
                 'flow_ttl_amt_1', 'flow_ttl_amt_drcr_ratio', 'active_days', 'acct_acno_in_num', 'acct_bank_no',
                 'user_id_level', 'req_lst_day', 'ori_req_brh_num', 'eb_check', 'mb_check', 'mb_limit']]
    #fillna
    data['eb_check'].fillna(value=0, inplace=True)
    data['mb_check'].fillna(value=0, inplace=True)
    data['mb_limit'].fillna(value=0, inplace=True)
    #Get columns
    columns = list(data.columns)
    drcr = pd.get_dummies(data['drcr'])
    drcr.rename(columns={1:'drcr_1', 2:'drcr_2'}, inplace=True)
    channel_desc_dummies = pd.get_dummies(data['channel_desc'])
    chal_1_dummies = pd.get_dummies(data['chal_1'])
    columns.remove('drcr')
    columns.remove('channel_desc')
    columns.remove('chal_1')
    x = data[columns]
    x = pd.concat([x, drcr, channel_desc_dummies, chal_1_dummies], axis=1)
    txn_model = pickle.load(open(txn_model_path, "rb"))
    col_list = ['tx_amt', 'pb_bal', 'time_diff', 'daily_total_1', 'daily_total_2',
                'amt_diff', 'tx_brh_cnt', 'income_withdraw_ratio', 'action_cnt',
                'session_total_amt', 'session_accumulated_amt', 'flow_total_amt_2',
                'flow_tx_amt_ratio', 'flow_tx_amt_seq', 'flow_tx_amt_seq_ratio',
                'flow_avg_time_diff', 'flow_ttl_amt_1', 'flow_ttl_amt_drcr_ratio',
                'active_days', 'acct_acno_in_num', 'acct_bank_no', 'user_id_level',
                'req_lst_day', 'ori_req_brh_num', 'drcr_1', 'drcr_2', 'ATM', 'Online',
                '網路銀行 網銀', '網路銀行 行銀', '銀行資訊交換平台FEP自行ATM', '銀行資訊交換平台FEP跨行ATM', 'eb_check', 'mb_check', 'mb_limit']
    for col in (set(col_list) - set(x.columns)):
        x[col] = 0
    proba = txn_model.predict_proba(x[col_list])
    return proba
```

選取交易模型建模所需欄位

補缺失值

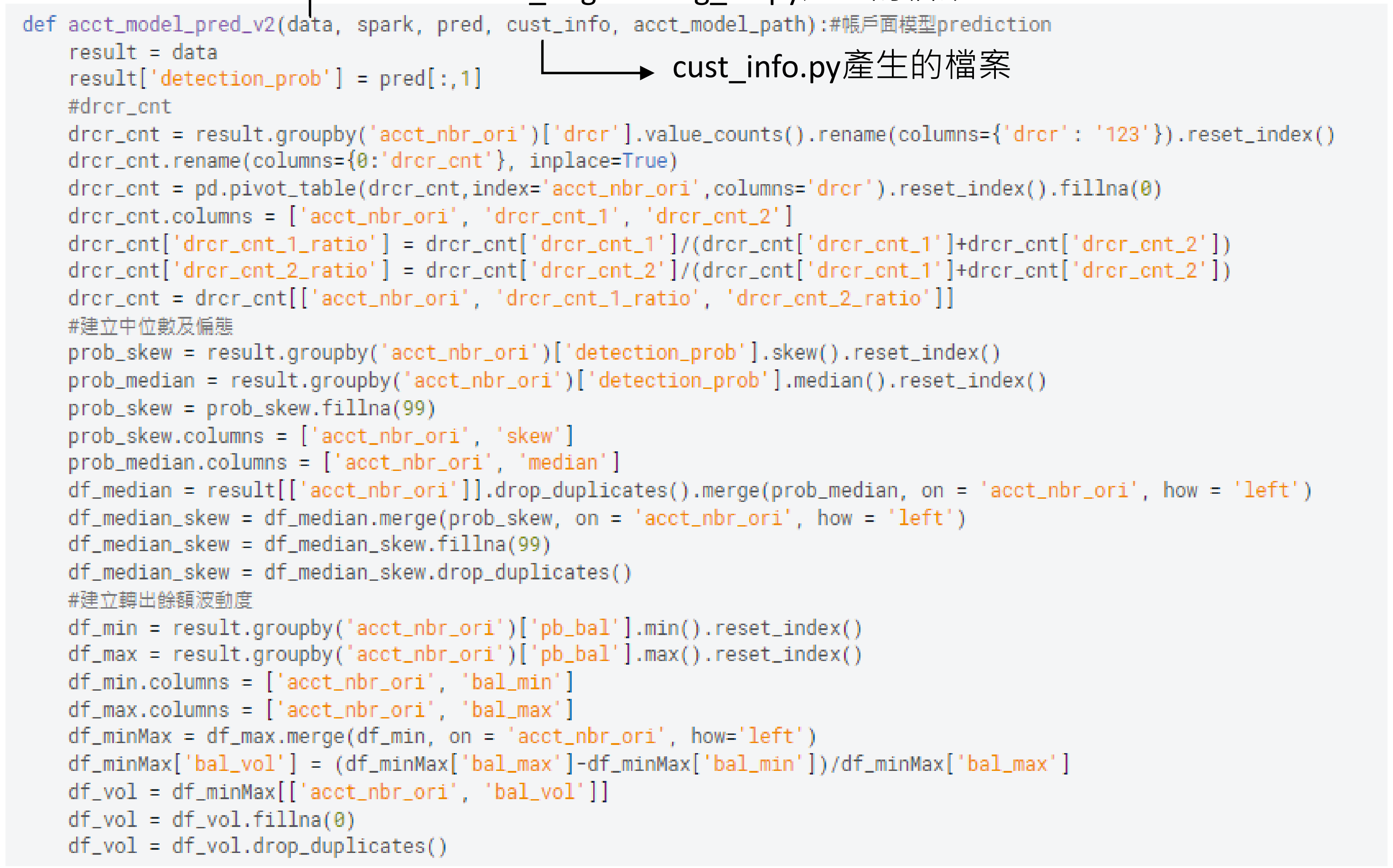
針對轉入/轉出、交易通路
建立虛擬變數

帳戶資料建立特徵

model_testing → acct_model_predict

feature_engineering_v2.py產生的檔案

cust_info.py產生的檔案



```
def acct_model_pred_v2(data, spark, pred, cust_info, acct_model_path):#帳戶面模型prediction
    result = data
    result['detection_prob'] = pred[:,1]
    #drcr_cnt
    drcr_cnt = result.groupby('acct_nbr_ori')['drcr'].value_counts().rename(columns={'drcr': '123'}).reset_index()
    drcr_cnt.rename(columns={0:'drcr_cnt'}, inplace=True)
    drcr_cnt = pd.pivot_table(drcr_cnt,index='acct_nbr_ori',columns='drcr').reset_index().fillna(0)
    drcr_cnt.columns = ['acct_nbr_ori', 'drcr_cnt_1', 'drcr_cnt_2']
    drcr_cnt['drcr_cnt_1_ratio'] = drcr_cnt['drcr_cnt_1']/(drcr_cnt['drcr_cnt_1']+drcr_cnt['drcr_cnt_2'])
    drcr_cnt['drcr_cnt_2_ratio'] = drcr_cnt['drcr_cnt_2']/(drcr_cnt['drcr_cnt_1']+drcr_cnt['drcr_cnt_2'])
    drcr_cnt = drcr_cnt[['acct_nbr_ori', 'drcr_cnt_1_ratio', 'drcr_cnt_2_ratio']]
    #建立中位數及偏態
    prob_skew = result.groupby('acct_nbr_ori')['detection_prob'].skew().reset_index()
    prob_median = result.groupby('acct_nbr_ori')['detection_prob'].median().reset_index()
    prob_skew = prob_skew.fillna(99)
    prob_skew.columns = ['acct_nbr_ori', 'skew']
    prob_median.columns = ['acct_nbr_ori', 'median']
    df_median = result[['acct_nbr_ori']].drop_duplicates().merge(prob_median, on = 'acct_nbr_ori', how = 'left')
    df_median_skew = df_median.merge(prob_skew, on = 'acct_nbr_ori', how = 'left')
    df_median_skew = df_median_skew.fillna(99)
    df_median_skew = df_median_skew.drop_duplicates()
    #建立轉出餘額波動度
    df_min = result.groupby('acct_nbr_ori')['pb_bal'].min().reset_index()
    df_max = result.groupby('acct_nbr_ori')['pb_bal'].max().reset_index()
    df_min.columns = ['acct_nbr_ori', 'bal_min']
    df_max.columns = ['acct_nbr_ori', 'bal_max']
    df_minMax = df_max.merge(df_min, on = 'acct_nbr_ori', how='left')
    df_minMax['bal_vol'] = (df_minMax['bal_max']-df_minMax['bal_min'])/df_minMax['bal_max']
    df_vol = df_minMax[['acct_nbr_ori', 'bal_vol']]
    df_vol = df_vol.fillna(0)
    df_vol = df_vol.drop_duplicates()
```

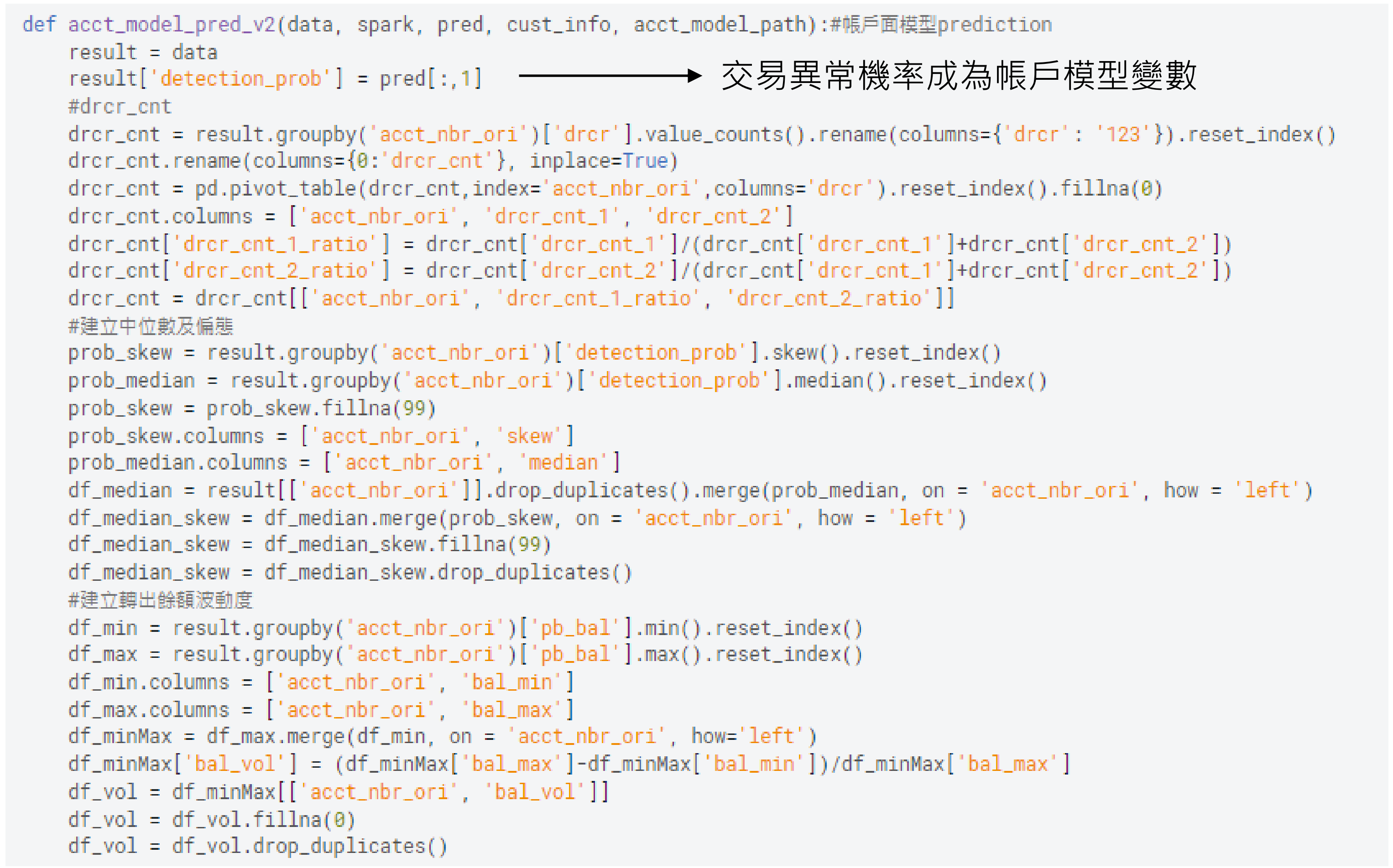
建立轉出次數、轉入/轉出次數比例

針對交易異常機率分布計算中位數及偏態

建立餘額波動度

帳戶資料建立特徵

model_testing → acct_model_predict



```python
def acct_model_pred_v2(data, spark, pred, cust_info, acct_model_path):#帳戶面模型prediction
    result = data
    result['detection_prob'] = pred[:,1]
    #drcr_cnt
    drcr_cnt = result.groupby('acct_nbr_ori')['drcr'].value_counts().rename(columns={'drcr': '123'}).reset_index()
    drcr_cnt.rename(columns={0:'drcr_cnt'}, inplace=True)
    drcr_cnt = pd.pivot_table(drcr_cnt,index='acct_nbr_ori',columns='drcr').reset_index().fillna(0)
    drcr_cnt.columns = ['acct_nbr_ori', 'drcr_cnt_1', 'drcr_cnt_2']
    drcr_cnt['drcr_cnt_1_ratio'] = drcr_cnt['drcr_cnt_1']/(drcr_cnt['drcr_cnt_1']+drcr_cnt['drcr_cnt_2'])
    drcr_cnt['drcr_cnt_2_ratio'] = drcr_cnt['drcr_cnt_2']/(drcr_cnt['drcr_cnt_1']+drcr_cnt['drcr_cnt_2'])
    drcr_cnt = drcr_cnt[['acct_nbr_ori', 'drcr_cnt_1_ratio', 'drcr_cnt_2_ratio']]
    #建立中位數及偏態
    prob_skew = result.groupby('acct_nbr_ori')['detection_prob'].skew().reset_index()
    prob_median = result.groupby('acct_nbr_ori')['detection_prob'].median().reset_index()
    prob_skew = prob_skew.fillna(99)
    prob_skew.columns = ['acct_nbr_ori', 'skew']
    prob_median.columns = ['acct_nbr_ori', 'median']
    df_median = result[['acct_nbr_ori']].drop_duplicates().merge(prob_median, on = 'acct_nbr_ori', how = 'left')
    df_median_skew = df_median.merge(prob_skew, on = 'acct_nbr_ori', how = 'left')
    df_median_skew = df_median_skew.fillna(99)
    df_median_skew = df_median_skew.drop_duplicates()
    #建立轉出餘額波動度
    df_min = result.groupby('acct_nbr_ori')['pb_bal'].min().reset_index()
    df_max = result.groupby('acct_nbr_ori')['pb_bal'].max().reset_index()
    df_min.columns = ['acct_nbr_ori', 'bal_min']
    df_max.columns = ['acct_nbr_ori', 'bal_max']
    df_minMax = df_max.merge(df_min, on = 'acct_nbr_ori', how='left')
    df_minMax['bal_vol'] = (df_minMax['bal_max']-df_minMax['bal_min'])/df_minMax['bal_max']
    df_vol = df_minMax[['acct_nbr_ori', 'bal_vol']]
    df_vol = df_vol.fillna(0)
    df_vol = df_vol.drop_duplicates()
```

→ 交易異常機率成為帳戶模型變數

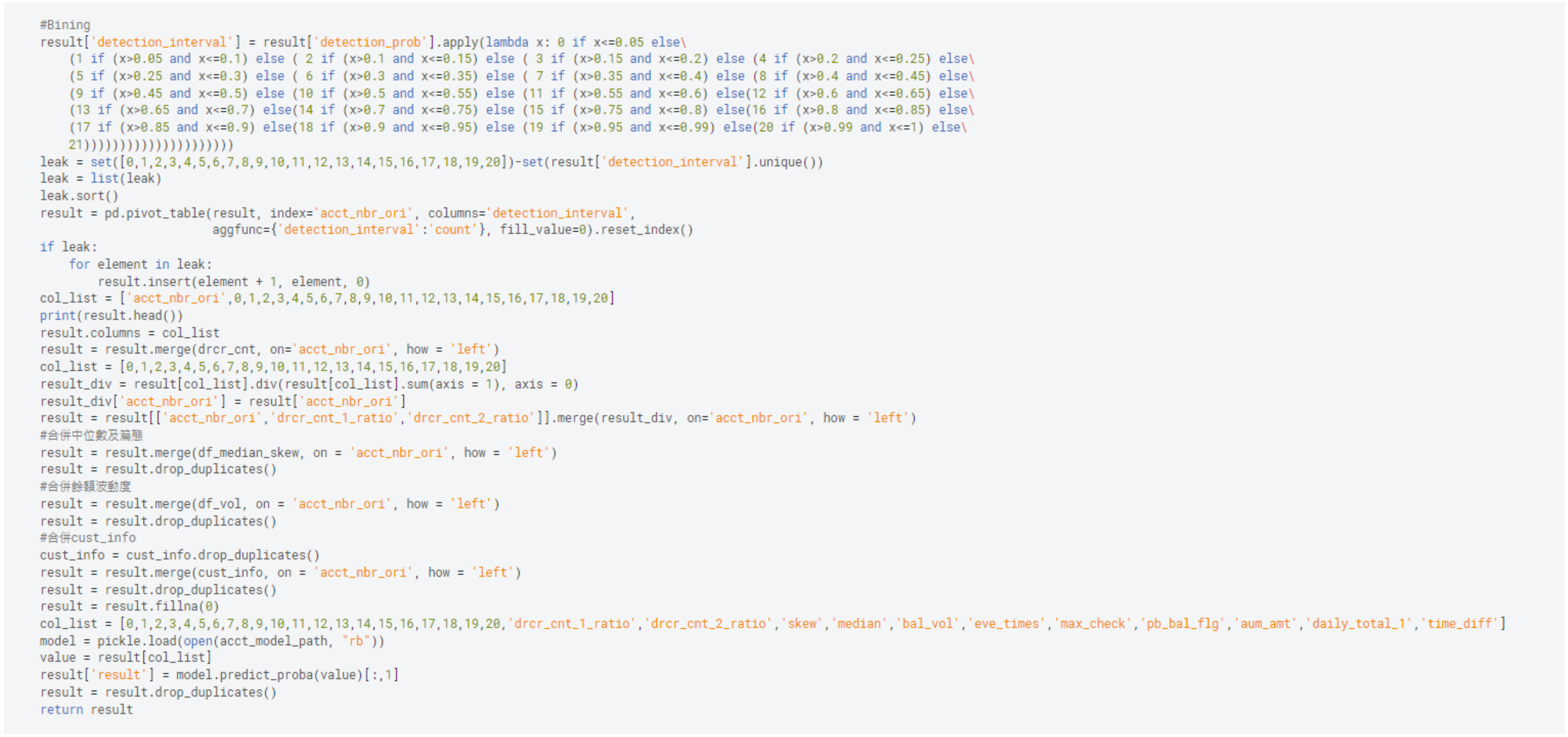
建立轉出次數、轉入/轉出次數比例

針對交易異常機率分布計算中位數及偏態

建立餘額波動度

帳戶資料建立特徵

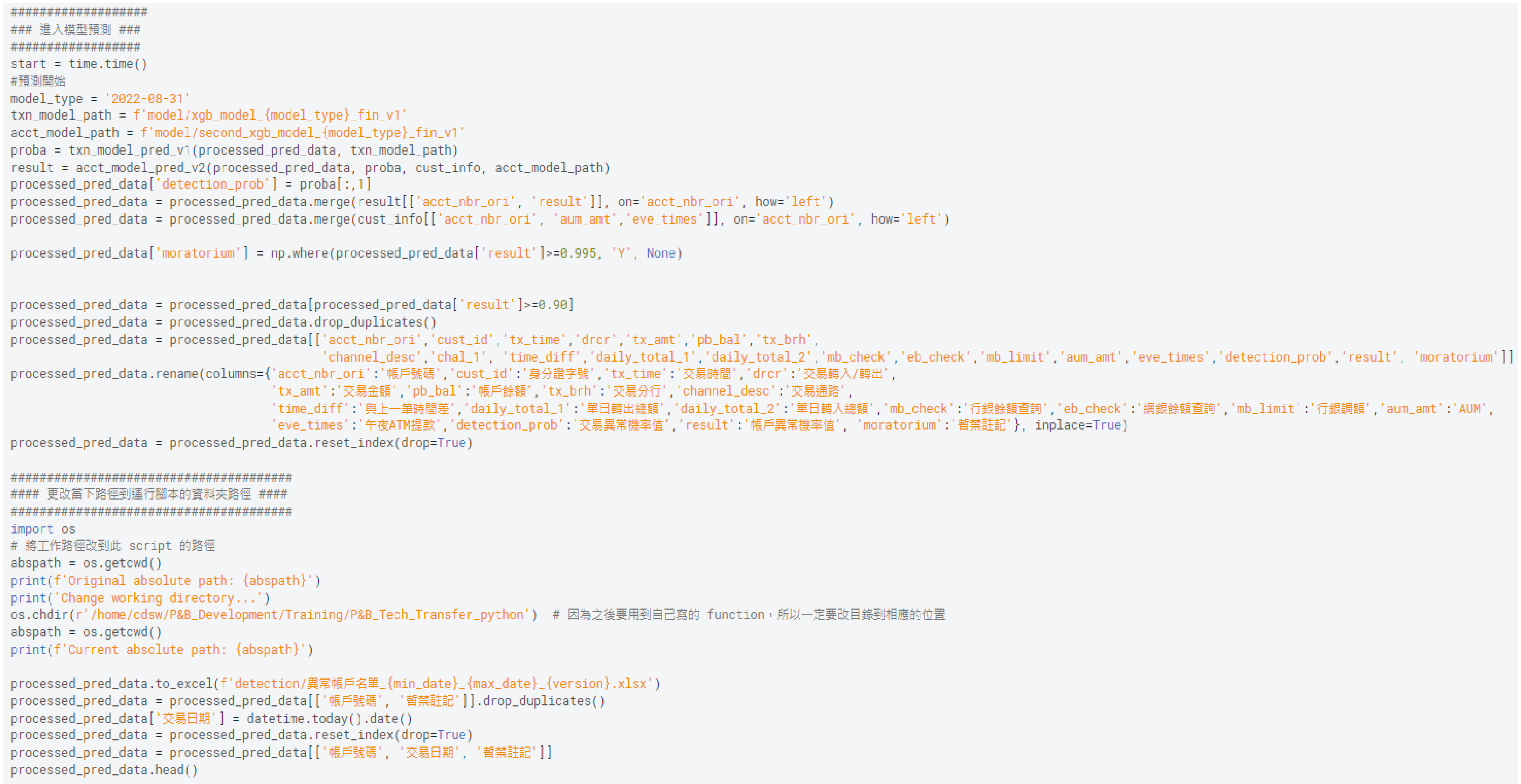
model_testing → acct_model_predict



```
#Bining
result['detection_interval'] = result['detection_prob'].apply(lambda x: 0 if x<=0.05 else\
    (1 if (x>0.05 and x<=0.1) else ( 2 if (x>0.1 and x<=0.15) else ( 3 if (x>0.15 and x<=0.2) else (4 if (x>0.2 and x<=0.25) else\
    (5 if (x>0.25 and x<=0.3) else ( 6 if (x>0.3 and x<=0.35) else ( 7 if (x>0.35 and x<=0.4) else (8 if (x>0.4 and x<=0.45) else\
    (9 if (x>0.45 and x<=0.5) else (10 if (x>0.5 and x<=0.55) else (11 if (x>0.55 and x<=0.6) else(12 if (x>0.6 and x<=0.65) else\
    (13 if (x>0.65 and x<=0.7) else(14 if (x>0.7 and x<=0.75) else (15 if (x>0.75 and x<=0.8) else(16 if (x>0.8 and x<=0.85) else\
    (17 if (x>0.85 and x<=0.9) else(18 if (x>0.9 and x<=0.95) else (19 if (x>0.95 and x<=0.99) else(20 if (x>0.99 and x<=1) else\
    21)))))))))))))))))))))
leak = set([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20])-set(result['detection_interval'].unique())
leak = list(leak)
leak.sort()
result = pd.pivot_table(result, index='acct_nbr_ori', columns='detection_interval',
                        aggfunc={'detection_interval':'count'}, fill_value=0).reset_index()
if leak:
    for element in leak:
        result.insert(element + 1, element, 0)
col_list = ['acct_nbr_ori',0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
print(result.head())
result.columns = col_list
result = result.merge(drcr_cnt, on='acct_nbr_ori', how = 'left')
col_list = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
result_div = result[col_list].div(result[col_list].sum(axis = 1), axis = 0)
result_div['acct_nbr_ori'] = result['acct_nbr_ori']
result = result[['acct_nbr_ori','drcr_cnt_1_ratio','drcr_cnt_2_ratio']].merge(result_div, on='acct_nbr_ori', how = 'left')
#合併中位數及偏態
result = result.merge(df_median_skew, on = 'acct_nbr_ori', how = 'left')
result = result.drop_duplicates()
#合併餘額波動度
result = result.merge(df_vol, on = 'acct_nbr_ori', how = 'left')
result = result.drop_duplicates()
#合併cust_info
cust_info = cust_info.drop_duplicates()
result = result.merge(cust_info, on = 'acct_nbr_ori', how = 'left')
result = result.drop_duplicates()
result = result.fillna(0)
col_list = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,'drcr_cnt_1_ratio','drcr_cnt_2_ratio','skew','median','bal_vol','eve_times','max_check','pb_bal_flg','aum_amt','daily_total_1','time_diff']
model = pickle.load(open(acct_model_path, "rb"))
value = result[col_list]
result['result'] = model.predict_proba(value)[:,1]
result = result.drop_duplicates()
return result
```

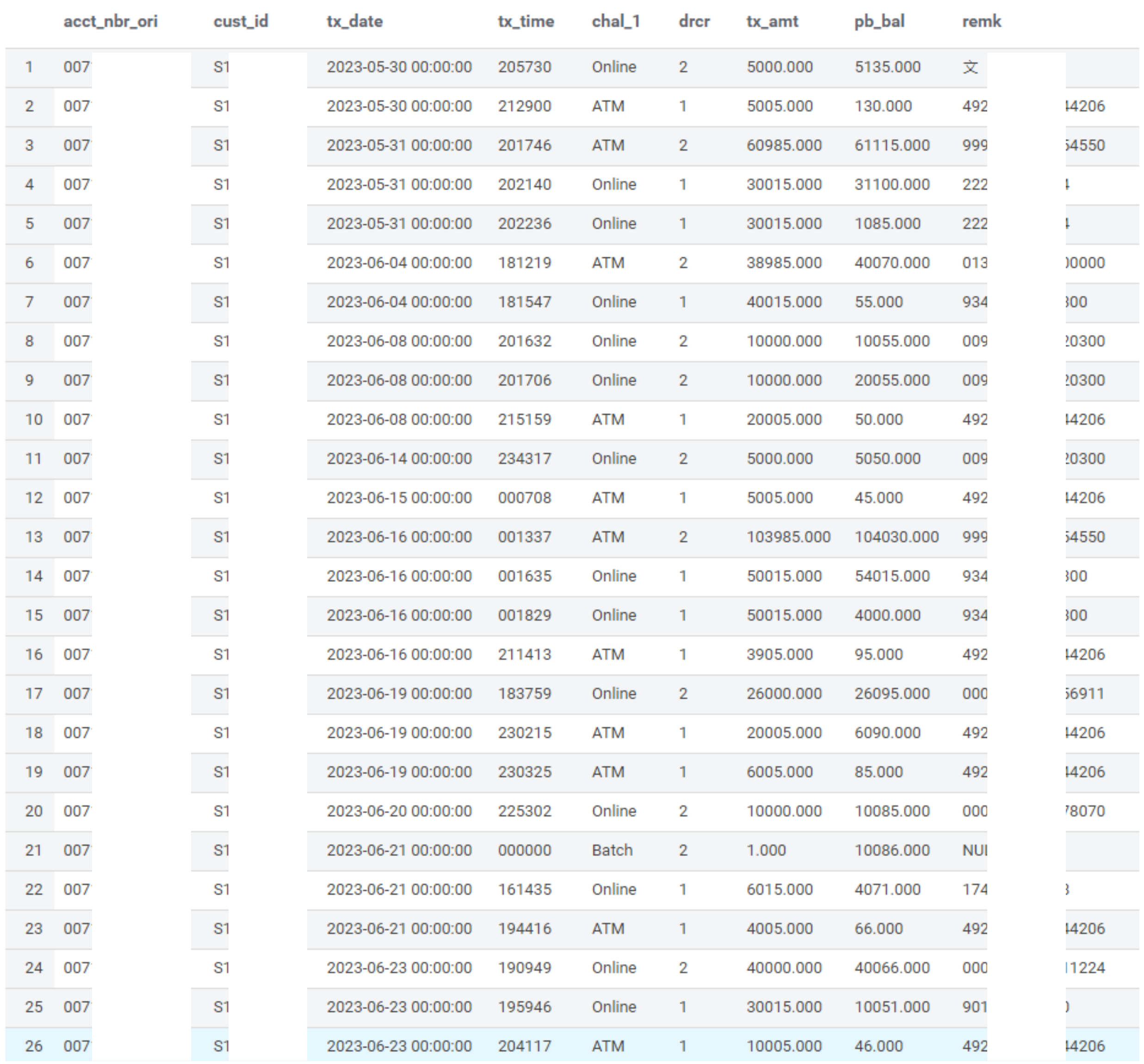
帳戶資料建立特徵

05.prediction_moratorium.py

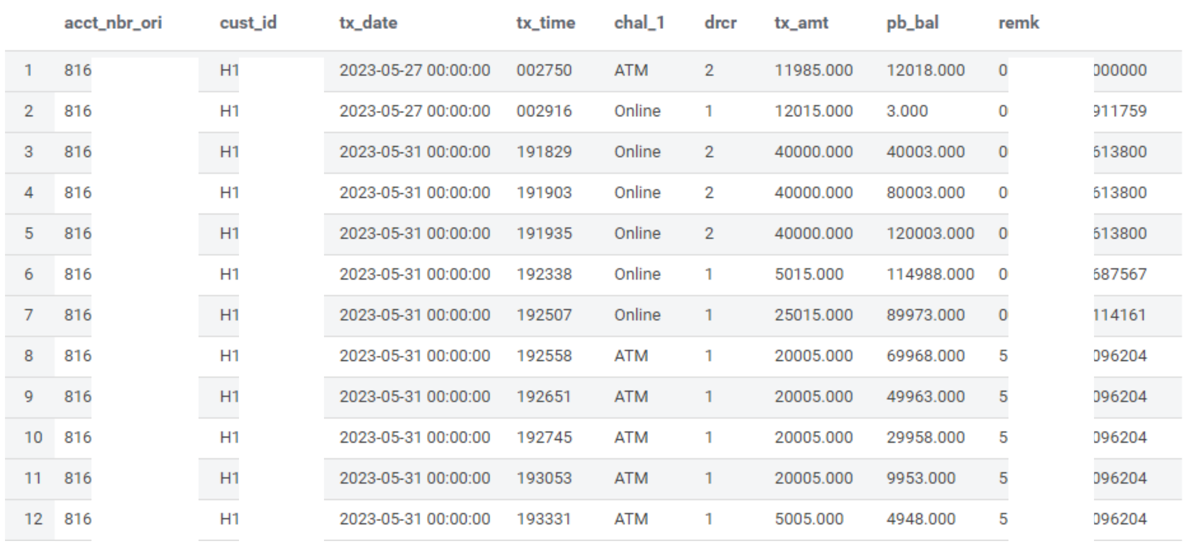


```
###################
### 進入模型預測 ###
##################
start = time.time()
#預測開始
model_type = '2022-08-31'
txn_model_path = f'model/xgb_model_{model_type}_fin_v1'
acct_model_path = f'model/second_xgb_model_{model_type}_fin_v1'
proba = txn_model_pred_v1(processed_pred_data, txn_model_path)
result = acct_model_pred_v2(processed_pred_data, proba, cust_info, acct_model_path)
processed_pred_data['detection_prob'] = proba[:,1]
processed_pred_data = processed_pred_data.merge(result[['acct_nbr_ori', 'result']], on='acct_nbr_ori', how='left')
processed_pred_data = processed_pred_data.merge(cust_info[['acct_nbr_ori', 'aum_amt','eve_times']], on='acct_nbr_ori', how='left')

processed_pred_data['moratorium'] = np.where(processed_pred_data['result']>=0.995, 'Y', None)


processed_pred_data = processed_pred_data[processed_pred_data['result']>=0.90]
processed_pred_data = processed_pred_data.drop_duplicates()
processed_pred_data = processed_pred_data[['acct_nbr_ori','cust_id','tx_time','drcr','tx_amt','pb_bal','tx_brh',
                                           'channel_desc','chal_1', 'time_diff','daily_total_1','daily_total_2','mb_check','eb_check','mb_limit','aum_amt','eve_times','detection_prob','result', 'moratorium']]
processed_pred_data.rename(columns={'acct_nbr_ori':'帳戶號碼','cust_id':'身分證字號','tx_time':'交易時間','drcr':'交易轉入/轉出',
                                    'tx_amt':'交易金額','pb_bal':'帳戶餘額','tx_brh':'交易分行','channel_desc':'交易通路',
                                    'time_diff':'與上一筆時間差','daily_total_1':'單日轉出總額','daily_total_2':'單日轉入總額','mb_check':'行銀餘額查詢','eb_check':'網銀餘額查詢','mb_limit':'行銀調額','aum_amt':'AUM',
                                    'eve_times':'午夜ATM提款','detection_prob':'交易異常機率值','result':'帳戶異常機率值', 'moratorium':'暂禁註記'}, inplace=True)
processed_pred_data = processed_pred_data.reset_index(drop=True)

#######################################
#### 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'/home/cdsw/P&B_Development/Training/P&B_Tech_Transfer_python')  # 因為之後要用到自己寫的 function，所以一定要改目錄到相應的位置
abspath = os.getcwd()
print(f'Current absolute path: {abspath}')

processed_pred_data.to_excel(f'detection/異常帳戶名單_{min_date}_{max_date}_{version}.xlsx')
processed_pred_data = processed_pred_data[['帳戶號碼', '暂禁註記']].drop_duplicates()
processed_pred_data['交易日期'] = datetime.today().date()
processed_pred_data = processed_pred_data.reset_index(drop=True)
processed_pred_data = processed_pred_data[['帳戶號碼', '交易日期', '暂禁註記']]
processed_pred_data.head()
```

預測結果



	acct_nbr_ori	cust_id	tx_date	tx_time	chal_1	drcr	tx_amt	pb_bal	remk	
1	007	S1	2023-05-30 00:00:00	205730	Online	2	5000.000	5135.000	文	
2	007	S1	2023-05-30 00:00:00	212900	ATM	1	5005.000	130.000	492	14206
3	007	S1	2023-05-31 00:00:00	201746	ATM	2	60985.000	61115.000	999	54550
4	007	S1	2023-05-31 00:00:00	202140	Online	1	30015.000	31100.000	222	
5	007	S1	2023-05-31 00:00:00	202236	Online	1	30015.000	1085.000	222	
6	007	S1	2023-06-04 00:00:00	181219	ATM	2	38985.000	40070.000	013	00000
7	007	S1	2023-06-04 00:00:00	181547	Online	1	40015.000	55.000	934	300
8	007	S1	2023-06-08 00:00:00	201632	Online	2	10000.000	10055.000	009	20300
9	007	S1	2023-06-08 00:00:00	201706	Online	2	10000.000	20055.000	009	20300
10	007	S1	2023-06-08 00:00:00	215159	ATM	1	20005.000	50.000	492	14206
11	007	S1	2023-06-14 00:00:00	234317	Online	2	5000.000	5050.000	009	20300
12	007	S1	2023-06-15 00:00:00	000708	ATM	1	5005.000	45.000	492	14206
13	007	S1	2023-06-16 00:00:00	001337	ATM	2	103985.000	104030.000	999	54550
14	007	S1	2023-06-16 00:00:00	001635	Online	1	50015.000	54015.000	934	300
15	007	S1	2023-06-16 00:00:00	001829	Online	1	50015.000	4000.000	934	300
16	007	S1	2023-06-16 00:00:00	211413	ATM	1	3905.000	95.000	492	14206
17	007	S1	2023-06-19 00:00:00	183759	Online	2	26000.000	26095.000	000	56911
18	007	S1	2023-06-19 00:00:00	230215	ATM	1	20005.000	6090.000	492	14206
19	007	S1	2023-06-19 00:00:00	230325	ATM	1	6005.000	85.000	492	14206
20	007	S1	2023-06-20 00:00:00	225302	Online	2	10000.000	10085.000	000	78070
21	007	S1	2023-06-21 00:00:00	000000	Batch	2	1.000	10086.000	NUI	
22	007	S1	2023-06-21 00:00:00	161435	Online	1	6015.000	4071.000	174	3
23	007	S1	2023-06-21 00:00:00	194416	ATM	1	4005.000	66.000	492	14206
24	007	S1	2023-06-23 00:00:00	190949	Online	2	40000.000	40066.000	000	11224
25	007	S1	2023-06-23 00:00:00	195946	Online	1	30015.000	10051.000	901	
26	007	S1	2023-06-23 00:00:00	204117	ATM	1	10005.000	46.000	492	14206



	acct_nbr_ori	cust_id	tx_date	tx_time	chal_1	drcr	tx_amt	pb_bal	remk	
1	816	H1	2023-05-27 00:00:00	002750	ATM	2	11985.000	12018.000	0	000000
2	816	H1	2023-05-27 00:00:00	002916	Online	1	12015.000	3.000	0	911759
3	816	H1	2023-05-31 00:00:00	191829	Online	2	40000.000	40003.000	0	613800
4	816	H1	2023-05-31 00:00:00	191903	Online	2	40000.000	80003.000	0	613800
5	816	H1	2023-05-31 00:00:00	191935	Online	2	40000.000	120003.000	0	613800
6	816	H1	2023-05-31 00:00:00	192338	Online	1	5015.000	114988.000	0	587567
7	816	H1	2023-05-31 00:00:00	192507	Online	1	25015.000	89973.000	0	114161
8	816	H1	2023-05-31 00:00:00	192558	ATM	1	20005.000	69968.000	5	096204
9	816	H1	2023-05-31 00:00:00	192651	ATM	1	20005.000	49963.000	5	096204
10	816	H1	2023-05-31 00:00:00	192745	ATM	1	20005.000	29958.000	5	096204
11	816	H1	2023-05-31 00:00:00	193053	ATM	1	20005.000	9953.000	5	096204
12	816	H1	2023-05-31 00:00:00	193331	ATM	1	5005.000	4948.000	5	096204

Q&A

Taipei Fubon Bank

Thank You!