To identify Good and Bad Accounts

In [1]:
```
cd D:\natxis
```

```
D:\natxis
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
#import quandl
import numpy as np
from tqdm import tqdm
import os
from collections import Counter

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    f1_score,
    accuracy_score,
    recall_score,
    precision_score,
    confusion_matrix,
    roc_auc_score,
    ConfusionMatrixDisplay,
    classification_report,
    precision_recall_curve
)
from sklearn import metrics
```

Import the training accounts

In [4]:
```
ta = pd.read_csv('train_accounts.csv')
```

Import the testing accounts

In [5]:
```
te = pd.read_csv('test_accounts.csv')
```

Import the transaction details

In [6]:
```
trans = pd.read_csv('transactions.csv')
```

In [7]:
```
trans.tail()
```

Out[7]:

	from_account	to_account	transaction_time_utc	value	gas	
5826599	b37259	a16395	2020-05-04 13:20:57	2000000000000000	21000	8(
5826600	a18542	b31501	2020-05-04 13:21:32	0	60000	10
5826601	a20151	b966524	2020-05-04 13:21:32	13000000000000000	21000	10
5826602	a25907	b31505	2020-05-04 13:22:10	0	1500000	12(
5826603	a20151	b31501	2020-05-04 13:22:10	0	60000	8

In [8]:
```
ta.tail()
```

Out[8]:

	account	flag
25193	a24443	0
25194	a12337	0
25195	a08122	0
25196	a27826	1
25197	a09863	1

In [9]:
```
te.tail()
```

Out[9]:

	account
6295	a19941
6296	a09327
6297	a10254
6298	a08928
6299	a03148

Profit of a transaction is determined by the difference between its value and the product of gas and gas price.

In [7]:
```
trans['pprofit'] = (trans.value.astype('float') - trans.gas*trans.gas_price)/100000
```

An indicator variable to show positive profit.

In [8]:
```
trans['profit'] = np.where(trans['pprofit'] > 0, trans['pprofit'],0)
```

Replace zero flag by -1 flag

In [9]:
```
ta.loc[ta[ta.flag==0].index,'flag']=-1
```

Merging the training and testing accounts

In [12]:
```
tdf = pd.merge(ta,te,left_on=['account'],right_on=['account'],how='outer')
```

In [13]:
```
tdf
```

Out[13]:

	account	flag
0	a17249	-1.0
1	a03683	1.0
2	a22146	-1.0
3	a26056	1.0
4	a13971	-1.0
...	...	...
31493	a19941	NaN
31494	a09327	NaN
31495	a10254	NaN
31496	a08928	NaN
31497	a03148	NaN

31498 rows × 2 columns

Since the testing account flags are missing, replace them with zero flags.

In [14]:
```
tdf.replace(np.nan,0,inplace=True)
```

Use account as index

In [15]:
```
tdf.set_index('account',inplace=True)
```

In [17]:
```
trans
```

Out[17]:

	from_account	to_account	transaction_time_utc	value	gas	
0	a00996	b31499	2020-05-04 14:54:03	0	72585	1
1	a07890	b31500	2020-05-04 14:55:06	0	54426	1
2	a22857	b31501	2020-05-04 14:55:23	0	200000	1
3	a07890	b31502	2020-05-04 14:55:23	108900000000000000	21000	1
4	a21390	b31501	2020-05-04 14:56:05	0	149999	3
...	...	...	...	...	...	
5826599	b37259	a16395	2020-05-04 13:20:57	2000000000000000	21000	
5826600	a18542	b31501	2020-05-04 13:21:32	0	60000	1
5826601	a20151	b966524	2020-05-04 13:21:32	13000000000000000	21000	1
5826602	a25907	b31505	2020-05-04 13:22:10	0	1500000	1
5826603	a20151	b31501	2020-05-04 13:22:10	0	60000	

5826604 rows × 8 columns

A function to find following transaction accounts

In [18]:
```
def find_to_nei(acc):
    neis = []
    s = trans[trans.from_account==acc]
    if len(s) > 0:
        neis.extend([(x,y,z) for x,y,z in zip(s.to_account.tolist(),s.profit.tolist
    return neis
```

In [19]:
```
find_to_nei('a26056')
```

Out[19]:
```
[('b394347', 2.230898581035051, 2.230898581035051),
 ('b394347', 2.3270455472120037, 2.3270455472120037),
 ('b404074', 0.023420912, 0.023420912),
 ('b502518', 0.03141765, 0.03141765),
 ('b399303', 0.04404585, 0.04404585),
 ('b394888', 0.572144449, 0.572144449)]
```

A function to find prior transaction accounts

In [20]:
```
def find_from_nei(acc):
    neis = []
    s = trans[trans.to_account==acc]
    if len(s) > 0:
        neis.extend([(x,y,z) for x,y,z in zip(s.from_account.tolist(),s.profit.toli
    return neis
```

In [21]:
```
find_from_nei('a26056')
```

Out[21]:
```
[('b31861', 0.049802, 0.049802),
 ('b503846', 0.0999685, 0.0999685),
 ('b503861', 0.199979, 0.199979),
 ('b31698', 0.05925, 0.05925),
 ('b31578', 0.09565127, 0.09565127),
 ('b417056', 0.123256994, 0.123256994),
 ('b31698', 0.04925, 0.04925),
 ('b450081', 0.049296205035051055, 0.049296205035051055),
 ('b504296', 0.04998992, 0.04998992),
 ('b504521', 0.18394719, 0.18394719),
 ('b31578', 0.048895, 0.048895),
 ('b504551', 0.0499916, 0.0499916),
 ('b504559', 0.1999916, 0.1999916),
 ('b33446', 0.248895, 0.248895),
 ('b487803', 0.3356765595, 0.3356765595),
 ('b31578', 0.059895, 0.059895),
 ('b504565', 0.0449937, 0.0449937),
 ('b397351', 0.059297255, 0.059297255),
 ('b31772', 0.00323416, 0.00323416),
 ('b31578', 0.0016293, 0.0016293),
 ('b31578', 0.004895, 0.004895),
 ('b33446', 0.050855429, 0.050855429),
 ('b31772', 0.056108804, 0.056108804),
 ('b504615', 0.0509559, 0.0509559),
 ('b504619', 0.050088354, 0.050088354),
 ('b31578', 0.07324318, 0.07324318),
 ('b33446', 0.049895, 0.049895),
 ('b498625', 0.049085, 0.049085),
 ('b504931', 0.055147899, 0.055147899),
 ('b504932', 0.4999937, 0.4999937),
 ('b31698', 0.04965, 0.04965),
 ('b504942', 0.0499874, 0.0499874),
 ('b504945', 0.0508194, 0.0508194),
 ('b31698', 0.49865, 0.49865),
 ('b505040', 0.0079916, 0.0079916),
 ('b32485', 0.0196960062120039, 0.0196960062120039),
 ('b505042', 0.0499559, 0.0499559),
 ('b505044', 0.0599916, 0.0599916),
 ('b31772', 0.051061671, 0.051061671),
 ('b505065', 0.059955861, 0.059955861),
 ('b505075', 0.0999916, 0.0999916),
 ('b505078', 0.0599895, 0.0599895),
 ('b36839', 0.4499538, 0.4499538),
 ('b31698', 0.08765, 0.08765),
 ('b417056', 0.049496, 0.049496),
 ('b505369', 0.049096, 0.049096),
 ('b505924', 0.343914261, 0.343914261),
 ('b506033', 0.049958, 0.049958),
 ('b389510', 0.01909, 0.01909),
 ('b506161', 0.0509937, 0.0509937),
 ('b506218', 0.0449139, 0.0449139),
 ('b31717', 0.0622945, 0.0622945)]
```

A function to find a forward transaction path of depth h

In [22]:
```
def find_forward_paths(acc,h):
    paths = [[(acc,0,0)]]
    if h > 0:
        stop = 0
        while stop == 0:
            newpaths = []
            for i in range(len(paths)):
                a_path = paths[i]
                u = a_path[-1][0]
                y = find_to_nei(u)
                if len(y) > 0:
                    for s,t,a in y:
                        x = a_path.copy()
                        x.append((s,t,a))
                        newpaths.append(x)
                else:
                    newpaths.append(a_path)
            if max(len(x) for x in newpaths) == h:
                stop = 1
            elif max(len(x) for x in newpaths) == max(len(x) for x in paths):
                stop = 1
            paths = newpaths
    return paths
```

cnt is a dataframe that summarizes the last transaction in a path if the path ends with an "a" account.

cnt2 is a dataframe that summarizes the last transaction in a path.

In [23]:
```
def find_weights(v):

    return output
```

A function to find a backward transaction path of depth h

In [24]:
```
def find_backward_paths(acc,h):

    return paths
```

v is a forward path that begins with 'a17249'

vv cnt summarizes the transactions that ends with 'a' account

vv cnt2 summarizes the transactions

In [25]:
```
v = find_forward_paths('a17249',3)
vv = find_weights(v)
```

In [26]:
```
vv['cnt'].groupby('flag').sum()
```

Out[26]:

flag	profit	pprofit	size
-1.0	0	0	1

In [27]:
```
vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[27]:

account	profit	pprofit	size
A	0	0	1

In [28]:
```
v = find_forward_paths('a25174',3)
vv = find_weights(v)
```

In [29]:
```
vv['cnt'].groupby('flag').sum()
```

Out[29]:

flag	profit	pprofit	size
-1.0	32960.050919	32960.050919	9072
0.0	20787.208347	20787.208347	4326
1.0	306.514683	306.514683	736

In [30]:
```
vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[30]:

account	profit	pprofit	size
A	54053.773948	54053.773948	14134
B	37.493935	37.485855	123

a11276 is a flag -1 account

vv cnt summarizes all forward transactions end with 'a' account. It shows that many transactions are with flag -1 accounts.

vv cnt2 summarizes all forward transactions. It shows that only 'a' accounts are involved.

In [31]:
```
#11276 - -1
v = find_forward_paths('a11276',3)
vv = find_weights(v)
```

In [32]:
```
w = find_backward_paths('a11276',3)
ww = find_weights(w)
```

In [33]:
```
vv['cnt'].groupby('flag').sum()
```

Out[33]:

flag	profit	pprofit	size
-1.0	123533.761061	123356.711198	203718
0.0	64606.696519	64604.749180	28462
1.0	6741.133469	6741.125424	14269

In [34]:
```
vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[34]:

account	profit	pprofit	size
A	194881.591049	194702.585803	246449

ww cnt summarizes all backward transactions. It shows that the number of transactions is much smaller.

In [35]:
```
ww['cnt'].groupby('flag').sum()
```

Out[35]:

flag	profit	pprofit	size
-1.0	24.249682	24.249082	1124
0.0	19.410232	19.406650	260
1.0	2.186452	2.186426	27

In [36]:
```
ww['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[36]:

account	profit	pprofit	size
A	45.846365	45.842158	1411
B	0.623779	0.623779	5

a09244 is a flag 1 account

vv cnt summarizes all forward transactions end with 'a' account. It shows that all transactions do not end with 'a' account.

vv cnt2 summarizes all forward transactions. It shows that the amount of transactions is very small.

In [37]:
```
#09244 - 1
v = find_forward_paths('a09244',3)
vv = find_weights(v)
```

In [38]:
```
w = find_backward_paths('a09244',3)
ww = find_weights(w)
```

In [39]:
```
vv
```

Out[39]:
```
{'cnt2':            profit   pprofit  size
 account
 b135242  0.034466  0.034466     1
 b141208  0.007203  0.007203     1}
```

In [41]:
```
vv['cnt'].groupby('flag').sum()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[41], line 1
----> 1 vv['cnt'].groupby('flag').sum()

KeyError: 'cnt'
```

In [42]:
```
vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[42]:

account	profit	pprofit	size
B	0.041669	0.041669	2

The number of backward transactions is also small.

In [43]:
```
ww['cnt'].groupby('flag').sum()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[43], line 1
----> 1 ww['cnt'].groupby('flag').sum()

KeyError: 'cnt'
```

In [44]:
```
ww['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[44]:

account	profit	pprofit	size
B	0.037952	0.037952	33

a26056 is a flag 1 account

vv cnt summarizes all forward transactions end with 'a' account. It shows that the number of transactions is small.

vv cnt2 summarizes all forward transactions. It shows that the number of transactions is small and most transactions are from 'a' accounts.

ww cnt summarizes all backward transactions that end with 'a' account. It shows that many transactions are with flag -1 accounts.

ww cnt2 summarizes all backward transactions. It shows that many transactions are with 'a' account.

In [340…
```
#26056 - 1
v = find_forward_paths('a26056',3)
vv = find_weights(v)
```

In [341…
```
w = find_backward_paths('a26056',3)
ww = find_weights(w)
```

In [342…
```
vv['cnt'].groupby('flag').sum()
```

Out[342…

flag	profit	size
-1.0	0.283069	2
0.0	2.441133	5
1.0	0.599956	1

In [343…
```
vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[343…

account	profit	size
A	3.324158	8
B	0.075463	2

In [344…
```
ww['cnt'].groupby('flag').sum()
```

Out[344...

flag	profit	size
-1.0	405731.865574	140611
0.0	103665.154172	42579
1.0	4075.162782	943

In [347...
```
ww['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[347...

account	profit	size
A	513472.182528	184133
B	2.880427	29

Account a23116, another flag 1 account

vv cnt summarizes all forward transactions end with 'a' account. It shows that no transactions ends with 'a' account.

vv cnt2 summarizes all forward transactions. It shows that the number of transactions is large and most transactions are from 'b' accounts.

ww cnt summarizes all backward transactions that end with 'a' account. It shows that many transactions are with flag -1 accounts.

ww cnt2 summarizes all backward transactions. It shows that many transactions are with 'a' account.

In [349...
```
#23116 - 1
v = find_forward_paths('a23116',3)
vv = find_weights(v)
```

In [350...
```
w = find_backward_paths('a23116',3)
ww = find_weights(w)
```

In [351...
```
vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[351...

account	profit	size
B	9.746286	21

In [352...
```
ww['cnt'].groupby('flag').sum()
```

Out[352...

flag	profit	size
-1.0	270719.635522	69804
0.0	68934.881538	21016
1.0	2721.122573	737

In [353...
```
ww['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[353...

account	profit	size
A	342375.639632	91557
B	11.148729	72

a03478 is a flag -1 account

vv cnt summarizes all forward transactions end with 'a' account. It shows that many of the transactions is with flag -1 account.

vv cnt2 summarizes all forward transactions. It shows that many transactions are 'a' accounts.

ww cnt summarizes all backward transactions end with 'a' account. It shows moderate number of transactions and mostly with flag -1 accounts

ww cnt2 summarizes all backward transactions. It shows that almost all transactions are 'a' account.

In [358...
```
#03478 - -1
v = find_forward_paths('a03478',3)
vv = find_weights(v)
```

In [359...
```
w = find_backward_paths('a03478',3)
ww = find_weights(w)
```

In [360...
```
vv['cnt'].groupby('flag').sum()
```

Out[360...

flag	profit	size
-1.0	71519.545878	117942
0.0	37403.876932	16478
1.0	3902.761482	8261

In [361…
```
vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[361…

account	profit	size
A	112826.184292	142681
B	0.000000	31

In [362…
```
ww['cnt'].groupby('flag').sum()
```

Out[362…

flag	profit	size
-1.0	235.677097	23930
0.0	60.278452	7128
1.0	1.287287	18

In [363…
```
ww['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[363…

account	profit	size
A	297.242836	31076
B	0.006044	1

a17547 is a flag -1 account

vv cnt summarizes all forward transactions end with 'a' account. It shows that there is only one transaction with flag -1 account.

vv cnt2 summarizes all forward transactions. It shows that most of the transactions are with 'b' accounts.

ww cnt summarizes all backward transactions end with 'a' account. It shows moderate number of transactions and mostly with flag -1 accounts

ww cnt2 summarizes all backward transactions. It shows that almost all transactions are 'a' account.

In [367…
```
#17547 - -1
v = find_forward_paths('a17547',3)
vv = find_weights(v)
```

In [368…
```
w = find_backward_paths('a17547',3)
```

```
ww = find_weights(w)
```

In [369…
```
vv['cnt'].groupby('flag').sum()
```

Out[369…

flag	profit	size
-1.0	0.009874	1

In [370…
```
vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[370…

account	profit	size
A	0.009874	1
B	1.964468	8

In [371…
```
ww['cnt'].groupby('flag').sum()
```

Out[371…

flag	profit	size
-1.0	67400.612860	16899
0.0	17216.473294	5221
1.0	676.203997	135

In [372…
```
ww['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[372…

account	profit	size
A	85293.290151	22255
B	0.134265	2

Account a23203, another flag -1 account

vv cnt summarizes all forward transactions end with 'a' account. It shows that there are small number of transactions ends with 'a' account and mostly with flag -1 account.

vv cnt2 summarizes all forward transactions. It shows that the number of transactions is mostly 'a' accounts.

ww cnt summarizes all backward transactions that end with 'a' account. It shows that there is no transaction end with 'a' account.

ww cnt2 summarizes all backward transactions. It shows that there are one transaction with 'b' account.

In [374…
```
#23203 - -1
v = find_forward_paths('a23203',3)
vv = find_weights(v)
```

In [375…
```
w = find_backward_paths('a23203',3)
ww = find_weights(w)
```

In [376…
```
vv['cnt'].groupby('flag').sum()
```

Out[376…

flag	profit	size
-1.0	2.316613	97
0.0	0.466151	6
1.0	0.032894	2

In [377…
```
vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[377…

account	profit	size
A	2.815658	105

In [378…
```
ww['cnt'].groupby('flag').sum()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\Users\KINGCH~1\AppData\Local\Temp/ipykernel_22188/3478351326.py in <module>
----> 1 ww['cnt'].groupby('flag').sum()

KeyError: 'cnt'
```

In [379…
```
ww['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[379…

account	profit	size
B	0.006993	1

Account a10607, another flag -1 account

vv cnt summarizes all forward transactions end with 'a' account. It shows that no transactions ends with 'a' account.

vv cnt2 summarizes all forward transactions. It shows that there are two transactions with 'b' accounts.

ww cnt summarizes all backward transactions that end with 'a' account. It shows that there is one transaction with flag -1 accounts.

ww cnt2 summarizes all backward transactions. It shows that the transaction is with 'a' account.

In [381...
```
#10607 - -1
v = find_forward_paths('a10607',3)
vv = find_weights(v)
```

In [382...
```
w = find_backward_paths('a10607',3)
ww = find_weights(w)
```

In [383...
```
vv['cnt'].groupby('flag').sum()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\Users\KINGCH~1\AppData\Local\Temp/ipykernel_22188/286493823.py in <module>
----> 1 vv['cnt'].groupby('flag').sum()

KeyError: 'cnt'
```

In [384...
```
vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[384...

account	profit	size
B	0.000396	2

In [385...
```
ww['cnt'].groupby('flag').sum()
```

Out[385...

flag	profit	size
-1.0	0.009832	1

In [386...
```
ww['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[386...

account	profit	size
A	0.009832	1

Account a03683, another flag 1 account

vv cnt summarizes all forward transactions end with 'a' account. It shows that no transactions ends with 'a' account.

vv cnt2 summarizes all forward transactions. It shows that there are three transactions with 'b' accounts.

ww cnt summarizes all backward transactions that end with 'a' account. It shows that no transactions ends with 'a' account.

ww cnt2 summarizes all backward transactions. It shows that there is one transaction are with 'b' account.

In [392...
```
#03683 - 1
v = find_forward_paths('a03683',3)
vv = find_weights(v)
```

In [398...
```
w = find_backward_paths('a03683',3)
ww = find_weights(w)
```

In [394...
```
vv['cnt'].groupby('flag').sum()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\Users\KINGCH~1\AppData\Local\Temp/ipykernel_22188/286493823.py in <module>
----> 1 vv['cnt'].groupby('flag').sum()

KeyError: 'cnt'
```

In [395...
```
vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[395...

account	profit	size
B	0.014059	3

In [399...
```
ww['cnt'].groupby('flag').sum()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\Users\KINGCH~1\AppData\Local\Temp/ipykernel_22188/3478351326.py in <module>
----> 1 ww['cnt'].groupby('flag').sum()

KeyError: 'cnt'
```

In [400...
```
ww['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
```

Out[400…

account	profit	size
B	0.006779	1

In [71]:
```
tdf.iloc[2000:2050]
```

Out[71]:

account	flag
a15154	-1.0
a00808	-1.0
a10127	-1.0
a06139	-1.0
a19666	-1.0
a24339	-1.0
a24894	-1.0
a25790	1.0
a07813	-1.0
a16078	-1.0
a09975	1.0
a19298	1.0
a03854	-1.0
a17818	-1.0
a06265	-1.0
a19650	-1.0
a09937	-1.0
a05831	-1.0
a11923	-1.0
a24466	-1.0
a26641	1.0
a05681	-1.0
a28314	-1.0
a22430	-1.0
a17733	-1.0
a21835	-1.0
a31105	-1.0
a08418	-1.0
a04911	-1.0

account	flag
a12591	-1.0
a00256	-1.0
a04957	-1.0
a21207	-1.0
a21386	-1.0
a28686	-1.0
a22219	-1.0
a16950	-1.0
a19787	1.0
a28039	-1.0
a28234	-1.0
a24477	-1.0
a11068	-1.0
a07174	-1.0
a12161	1.0
a06278	-1.0
a20827	-1.0
a28348	1.0
a21213	-1.0
a14727	-1.0
a12684	-1.0

In [301…
```
tdf[tdf.index=='a09360']
```

Out[301…

account	flag
a09360	-1.0

In [322…
```
#17808 - -1
v = find_forward_paths('a04276',3)
w = find_backward_paths('a04276',3)
vv = find_weights(v)
ww = find_weights(w)
```

In [323…
```
U = vv['cnt'].groupby('flag').sum()
U
```

Out[323…

flag	profit	pprofit	size
-1.0	221.361308	221.361308	2046
0.0	308.265462	308.265352	2156
1.0	36.328790	36.320324	737

In [324…
```
V = vv['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
V
```

Out[324…

account	profit	pprofit	size
A	565.95556	565.946984	4939

In [259…
```
V.loc['B','profit']*V.loc['B','size']
```

Out[259…
```
12.654163200000001
```

In [325…
```
ww['cnt'].groupby('flag').sum()
```

Out[325…

flag	profit	pprofit	size
-1.0	67400.648459	67400.648459	16901
0.0	17216.473294	17216.473255	5219
1.0	676.203997	676.203997	135

In [326…
```
W = ww['cnt2'].groupby(lambda x: 'A' if x[0]=='a' else 'B').sum()
W
```

Out[326…

account	profit	pprofit	size
A	85293.325751	85293.325712	22255
B	0.129850	0.129850	5

In [408…
```
W.loc['B','profit']*W.loc['B','size']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, meth
od, tolerance)
   3360             try:
-> 3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngi
ne.get_loc()

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngi
ne.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.
get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.
get_item()

KeyError: 'B'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
C:\Users\KINGCH~1\AppData\Local\Temp/ipykernel_22188/3988401894.py in <module>
----> 1 W.loc['B','profit']*W.loc['B','size']

~\anaconda3\lib\site-packages\pandas\core\indexing.py in __getitem__(self, key)
    923                 with suppress(KeyError, IndexError):
    924                     return self.obj._get_value(*key, takeable=self._takeabl
e)
--> 925             return self._getitem_tuple(key)
    926         else:
    927             # we by definition only have the 0th axis

~\anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_tuple(self, tup)
   1098     def _getitem_tuple(self, tup: tuple):
   1099         with suppress(IndexingError):
-> 1100             return self._getitem_lowerdim(tup)
   1101
   1102         # no multi-index, so validate all of the indexers

~\anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_lowerdim(self, tu
p)
    836                 # We don't need to check for tuples here because those are
    837                 #  caught by the _is_nested_tuple_indexer check above.
--> 838                 section = self._getitem_axis(key, axis=i)
    839
    840                 # We should never have a scalar section here, because

~\anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_axis(self, key, ax
is)
   1162         # fall thru to straight lookup
   1163         self._validate_key(key, axis)
-> 1164         return self._get_label(key, axis=axis)
```

```
   1165
   1166     def _get_slice_axis(self, slice_obj: slice, axis: int):

~\anaconda3\lib\site-packages\pandas\core\indexing.py in _get_label(self, label, axis)
   1111     def _get_label(self, label, axis: int):
   1112         # GH#5667 this will fail if the label is not present in the axis.
-> 1113         return self.obj.xs(label, axis=axis)
   1114
   1115     def _handle_lowerdim_multi_index_axis0(self, tup: tuple):

~\anaconda3\lib\site-packages\pandas\core\generic.py in xs(self, key, axis, level, drop_level)
   3774                 raise TypeError(f"Expected label or tuple of labels, got {key}") from e
   3775         else:
-> 3776             loc = index.get_loc(key)
   3777
   3778             if isinstance(loc, np.ndarray):

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:
-> 3363                 raise KeyError(key) from err
   3364
   3365         if is_scalar(key) and isna(key) and not self.hasnans:

KeyError: 'B'
```

In [306…
```
U = vv['cnt'].groupby('flag').sum()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\Users\KINGCH~1\AppData\Local\Temp/ipykernel_22188/2798145386.py in <module>
----> 1 U = vv['cnt'].groupby('flag').sum()

KeyError: 'cnt'
```

In [152…
```
U
```

Out[152…

flag	profit	size
-1.0	0.0	1

In [163…
```
'cnt' in vv
```

Out[163… False

In [422…
```
'B' in V.index
```

Out[422… True

In [164…
```
V.loc['B','profit'] > 0
```

Out[164…
```
True
```

Strategy is to divide the transactions into one of the four types

1. cnt exists in vv and cnt exists in ww
2. cnt exists in vv and cnt does not exist in ww
3. cnt does not exist in vv and cnt exists in ww
4. cnt does not exist in vv and cnt does not exist in ww

We build a random forest model to each case.

In [36]:
```
data1_df = pd.read_csv('data1.csv')
data1_df.drop(['Unnamed: 0'],axis=1,inplace=True)
len(data1_df)
```

Out[36]:
```
5029
```

In [37]:
```
data1_df.tail()
```

Out[37]:

	account	normal_fprofit	normal_fpprofit	normal_fsize	abnormal_fprofit	abnormal_f
5024	a17386	0.067153	0.067153	21	0.000000	0.
5025	a16265	0.001343	0.001343	1	0.000000	0.
5026	a16253	55.811596	55.811596	104	8.370918	8.
5027	a09866	0.068724	0.068724	4	0.000000	0.
5028	a13649	0.201987	0.201987	2	0.000000	0.

5 rows × 25 columns

In [38]:
```
data1 = {}
```

In [40]:
```
data2_df = pd.read_csv('data2.csv')
data2_df.drop(['Unnamed: 0'],axis=1,inplace=True)
len(data2_df)
```

Out[40]:
```
2365
```

In [41]:
```
data2_df.tail()
```

Out[41]:

	account	normal_fprofit	normal_fpprofit	normal_fsize	abnormal_fprofit	abnormal_f
2360	a29807	0.000392	0.000392	3	0.000000	0.
2361	a14441	0.000000	-0.000006	1	0.000000	0.
2362	a00340	1.289279	1.289279	56	0.000000	0.
2363	a00933	39010.661388	38954.750905	64332	2128.778990	2128.
2364	a25329	227562.191429	227236.046944	375270	12417.877443	12417.

In [42]:
```
data2 = {}
```

In [43]:
```
data3_df = pd.read_csv('data3.csv')
data3_df.drop(['Unnamed: 0'],axis=1,inplace=True)
len(data3_df)
```

Out[43]: 2326

In [44]:
```
data3_df.tail()
```

Out[44]:

	account	A_fprofit	A_fpprofit	A_fsize	B_fprofit	B_fpprofit	B_fsize	normal_bprofit
2321	a05614	0	0	0	0.034123	0.031651	27	20.738679
2322	a04941	0	0	0	0.723788	0.723788	11	0.009272
2323	a11441	0	0	0	30.099304	30.099304	5	0.005956
2324	a24707	0	0	0	0.032531	0.032531	5	0.000000
2325	a25213	0	0	0	0.248708	0.248708	2	67400.612860

In [45]:
```
data3 = {}
```

In [46]:
```
data4_df = pd.read_csv('data4.csv')
data4_df.drop(['Unnamed: 0'],axis=1,inplace=True)
len(data4_df)
```

Out[46]: 4577

In [47]:
```
data4_df.tail()
```

Out[47]:

	account	A_fprofit	A_fpprofit	A_fsize	B_fprofit	B_fpprofit	B_fsize	A_bprofit	A_b
4572	a29084	0	0	0	0.115200	0.115200	9	0	
4573	a14615	0	0	0	3.357953	3.357953	38	0	
4574	a22988	0	0	0	0.537370	0.537370	1	0	
4575	a10453	0	0	0	0.017810	0.017810	4	0	
4576	a29173	0	0	0	0.000000	-0.002129	57	0	

In [48]:
```
data4 = {}
```

In [49]:
```
data1 = {'account':[],'normal_fprofit':[],'normal_fpprofit':[],'normal_fsize':[],
         'abnormal_fprofit':[],'abnormal_fpprofit':[],'abnormal_fsize':[],
         'A_fprofit':[],'A_fpprofit':[],'A_fsize':[],'B_fprofit':[],'B_fpprofit':[]
         'normal_bprofit':[],'normal_bpprofit':[],'normal_bsize':[],
         'abnormal_bprofit':[],'abnormal_bpprofit':[],'abnormal_bsize':[],
         'A_bprofit':[],'A_bpprofit':[],'A_bsize':[],'B_bprofit':[],'B_bpprofit':[]
data2 = {'account':[],'normal_fprofit':[],'normal_fpprofit':[],'normal_fsize':[],
         'abnormal_fprofit':[],'abnormal_fpprofit':[],'abnormal_fsize':[],
         'A_fprofit':[],'A_fpprofit':[],'A_fsize':[],'B_fprofit':[],'B_fpprofit':[]
         'A_bprofit':[],'A_bpprofit':[],'A_bsize':[],'B_bprofit':[],'B_bpprofit':[]
data3 = {'account':[],'A_fprofit':[],'A_fpprofit':[],'A_fsize':[],'B_fprofit':[],'B
         'normal_bprofit':[],'normal_bpprofit':[],'normal_bsize':[],
         'abnormal_bprofit':[],'abnormal_bpprofit':[],'abnormal_bsize':[],
         'A_bprofit':[],'A_bpprofit':[],'A_bsize':[],'B_bprofit':[],'B_bpprofit':[]
data4 = {'account':[],'A_fprofit':[],'A_fpprofit':[],'A_fsize':[],'B_fprofit':[],'B
         'A_bprofit':[],'A_bpprofit':[],'A_bsize':[],'B_bprofit':[],'B_bpprofit':[]
for i in tqdm(range(len(ta))):
```

```
100%|████████████████████████████████████████████████████████████████| 1090
1/10901 [135:33:22<00:00, 44.77s/it]
```

In [54]:
```
ta
```

Out[54]:

	account	flag
0	a17249	-1
1	a03683	1
2	a22146	-1
3	a26056	1
4	a13971	-1
...	...	...
25193	a24443	-1
25194	a12337	-1
25195	a08122	-1
25196	a27826	1
25197	a09863	1

25198 rows × 2 columns

In [58]:
```
data1 = pd.DataFrame(data1)
data2 = pd.DataFrame(data2)
data3 = pd.DataFrame(data3)
data4 = pd.DataFrame(data4)
```

In [59]:
```
data1['flag'] = 0
data2['flag'] = 0
data3['flag'] = 0
data4['flag'] = 0
acc1 = data1.account.tolist()
acc2 = data2.account.tolist()
acc3 = data3.account.tolist()
acc4 = data4.account.tolist()
for i in range(len(ta)):
    if ta.account[i] in acc1:
        s = data1[data1.account==ta.account[i]].index
        data1.loc[s,'flag'] = ta.loc[i,'flag']
    if ta.account[i] in acc2:
        s = data2[data2.account==ta.account[i]].index
        data2.loc[s,'flag'] = ta.loc[i,'flag']
    if ta.account[i] in acc3:
        s = data3[data3.account==ta.account[i]].index
        data3.loc[s,'flag'] = ta.loc[i,'flag']
    if ta.account[i] in acc4:
        s = data4[data4.account==ta.account[i]].index
        data4.loc[s,'flag'] = ta.loc[i,'flag']
```

In [63]:
```
data4.groupby('flag').size()
```

Out[63]:
```
flag
-1    7445
 1     677
dtype: int64
```

In [ ]:
```
idx = data4[data4.flag==-1].index
data4.loc[idx,'flag'] = 0
```

In [127…
```
y_preds4 = []
for i in range(100):
    train41 = data4[data4.flag==1].sample(474)
    train40 = data4[data4.flag==0].sample(474)
    train4 = pd.concat([train40,train41],axis='rows')
    x_train4 = train4[train4.columns[1:-1]].values
    y_train4 = train4.flag.values
    clf4=RandomForestClassifier(n_estimators=100)
    clf4.fit(x_train4,y_train4)
    y_preds4.append(clf4.predict(data4[data4.columns[1:-1]].values))
```

In [128…
```
y_pred4 = np.where(pd.DataFrame(y_preds4).sum(axis='rows').values > 93,1,0)
y_pred4
```

Out[128…
```
array([0, 0, 0, ..., 0, 0, 1])
```

In [129…
```
print("Accuracy:",metrics.accuracy_score(data4.flag.values, y_pred4))
```

```
Accuracy: 0.9647869982762867
```

In [65]:
```
data3.groupby('flag').size()
```

Out[65]:
```
flag
-1    3309
 1     764
dtype: int64
```

In [126…
```
idx = data3[data3.flag==-1].index
data3.loc[idx,'flag'] = 0
```

In [130…
```
y_preds3 = []
for i in range(100):
    train31 = data3[data3.flag==1].sample(535)
    train30 = data3[data3.flag==0].sample(535)
    train3 = pd.concat([train30,train31],axis='rows')
    x_train3 = train3[train3.columns[1:-1]].values
    y_train3 = train3.flag.values
    clf3=RandomForestClassifier(n_estimators=100)
    clf3.fit(x_train3,y_train3)
    y_preds3.append(clf3.predict(data3[data3.columns[1:-1]].values))
```

In [131…
```
y_pred3 = np.where(pd.DataFrame(y_preds3).sum(axis='rows').values > 93,1,0)
y_pred3
```

Out[131…
```
array([0, 0, 1, ..., 0, 0, 0])
```

In [132…
```
print("Accuracy:",metrics.accuracy_score(data3.flag.values, y_pred3))
```

```
Accuracy: 0.9543334151730911
```

In [133…
```
data2.groupby('flag').size()
```

Out[133…
```
flag
-1    3912
 1     254
dtype: int64
```

In [134…
```
idx = data2[data2.flag==-1].index
data2.loc[idx,'flag'] = 0
```

In [135…
```
y_preds2 = []
for i in range(100):
    train21 = data2[data2.flag==1].sample(178)
    train20 = data2[data2.flag==0].sample(178)
    train2 = pd.concat([train20,train21],axis='rows')
    x_train2 = train2[train2.columns[1:-1]].values
    y_train2 = train2.flag.values
    clf2=RandomForestClassifier(n_estimators=100)
    clf2.fit(x_train2,y_train2)
    y_preds2.append(clf2.predict(data2[data2.columns[1:-1]].values))
```

In [136…
```
y_pred2 = np.where(pd.DataFrame(y_preds2).sum(axis='rows').values > 93,1,0)
y_pred2
```

Out[136…
```
array([0, 0, 0, ..., 0, 1, 0])
```

In [137…
```
print("Accuracy:",metrics.accuracy_score(data2.flag.values, y_pred2))
```

```
Accuracy: 0.9827172347575612
```

In [139…
```
data1.groupby('flag').size()
```

Out[139…
```
flag
0    8077
1     760
dtype: int64
```

In [138…
```
idx = data1[data1.flag==-1].index
data1.loc[idx,'flag'] = 0
```

In [140…
```
y_preds1 = []
for i in range(100):
    train11 = data1[data1.flag==1].sample(532)
    train10 = data1[data1.flag==0].sample(532)
    train1 = pd.concat([train10,train11],axis='rows')
    x_train1 = train1[train1.columns[1:-1]].values
    y_train1 = train1.flag.values
    clf1=RandomForestClassifier(n_estimators=100)
    clf1.fit(x_train1,y_train1)
    y_preds1.append(clf1.predict(data1[data1.columns[1:-1]].values))
```

In [141…
```
y_pred1 = np.where(pd.DataFrame(y_preds1).sum(axis='rows').values > 93,1,0)
y_pred1
```

Out[141…
```
array([0, 1, 0, ..., 0, 0, 1])
```

In [142…
```
print("Accuracy:",metrics.accuracy_score(data1.flag.values, y_pred1))
```

```
Accuracy: 0.9861944098676021
```

In [ ]: