C

1 算法阐述

考虑到每个的交易情况，选取aii中使用pluno得到的频繁集。

若某个频繁集减去这个频繁集与某个用户历史交易中的items只剩下一个item，则将这个item推荐给该用户。

对该用户被推荐的所有items与后40%交易中该用户交易的item进行准确率分析。

$$Precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ document\}|}{|\{retrieved\ ducoments\}|} = \frac{tn}{fp+tp}$$

$$Recall = \frac{|\{relevant\ documents\} \cap \{retrieved\ document\}|}{|\{retrieved\ ducoments\}|} = \frac{tp}{tp+fn}$$

代码

```
old_seq = trade_new_df[['dptno', 'vipno',
'sldatime']].groupby('vipno').apply(
    lambda x: x.sort_values(by='sldatime', ascending=False).head(
        int(x['dptno'].count() * 0.6))).groupby('vipno')['dptno'].apply(
    lambda x: x.tolist()).to_dict()

new_seq = trade_new_df[['dptno', 'vipno',
'sldatime']].groupby('vipno').apply(
        lambda x: x.sort_values(by='sldatime', ascending=False).head(
            int(x['dptno'].count() * 0.4))).groupby(['vipno', 'sldatime'])
['dptno'].apply(
        lambda x: x.tolist()).to_dict()

reco = {}
precision = {}
recall = {}
for pattern in trade_new_dptno_fp:
    for vipno, old_tran in old_seq.items():
        remaining = {*pattern} - {*pattern}.intersection(old_tran)
        if len(remaining) == 1 and len(pattern) > 1:
            reco[vipno] = reco.get(vipno, set())
            reco[vipno].add(lambda x: x in remaining)
for index, new_tran in new_seq.items():
    vipno = index[0]
    hit = len(set(reco[vipno]).intersection(new_tran))
    precision[vipno] = hit / len(new_tran)
    recall[vipno] = hit / len(reco[vipno])
```

```
print('precision: {}'.format(mean(precision.values())))
print('recall: {}'.format(mean(recall.values())))
```

2 讨论分析

support	10	8	6	4	2
precision	0.1241	0.1137	0.1323	0.1791	0.0423
recall	0.0541	0.0724	0.0859	0.0817	0.0237

该算法在各个`support`下的`Precision`和`recall`都很低，表明该预测算法在此业务下不适用。

3 性能