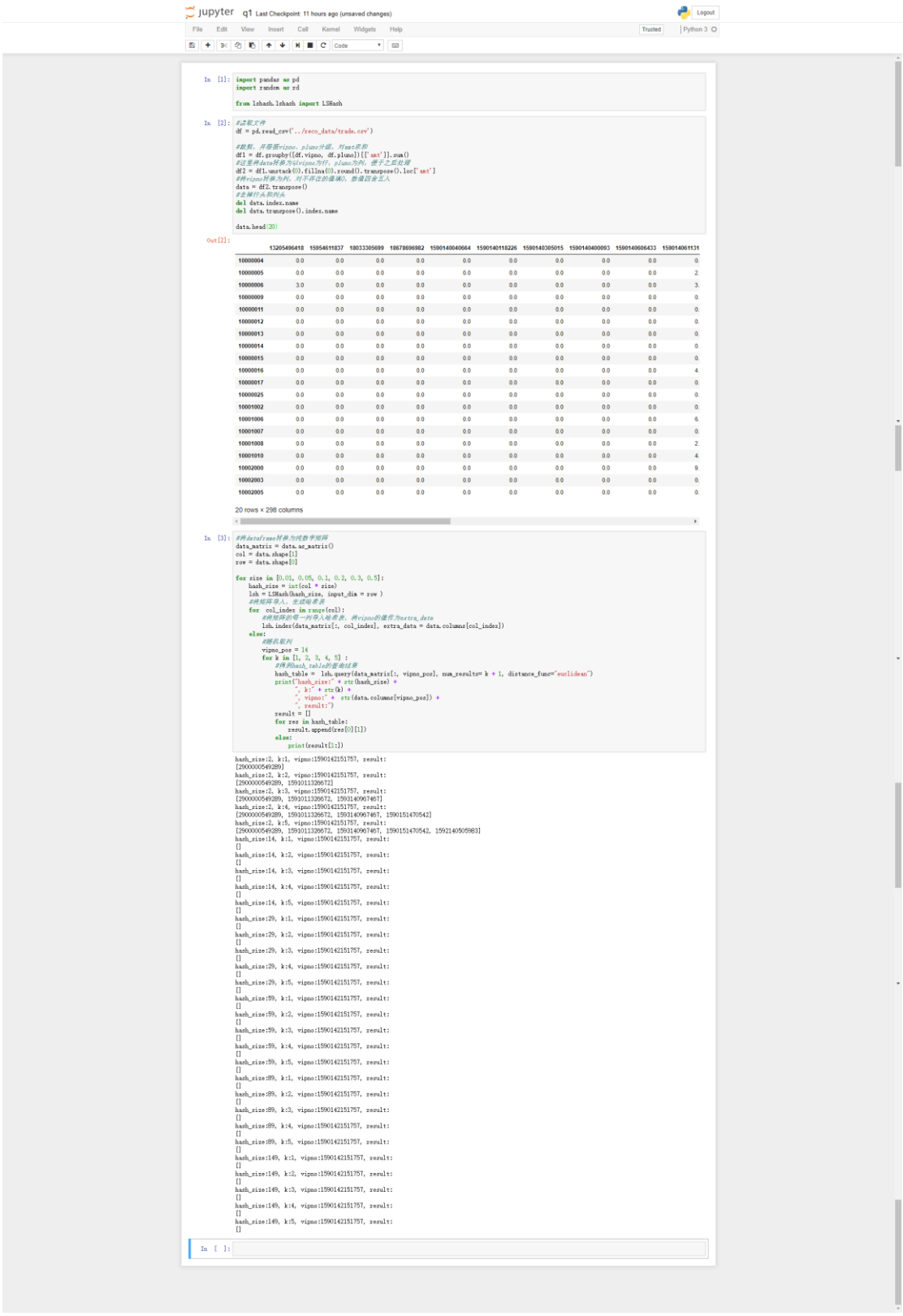
Quetion1 报告

一、运行结果



jupyter q1 Last Checkpoint: 11 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Logout | Trusted | Python 3

In [1]:
```
import pandas as pd
import random as rd

from lshash.lshash import LSHash
```

In [2]:
```
#读取文件
df = pd.read_csv('../reco_data/trade.csv')

#数据，并按照vipno、pluno分组，对amt求和
df1 = df.groupby([df.vipno, df.pluno])[['amt']].sum()
#过里再data转换为以vipno为行，pluno为列，便于之后处理
df2 = df1.unstack(0).fillna(0).round().transpose().loc['amt']
#将vipno转换为列，对不存在的值填0，数值四舍五入
data = df2.transpose()
#去掉行头和列头
del data.index.name
del data.transpose().index.name

data.head(20)
```

Out[2]:

	1320549418	15954611837	18033305699	18678696982	1590140040664	1590140118226	1590140305015	1590140400093	1590140606433	1590140611311
10000004	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10000005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.
10000006	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.
10000009	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10000011	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10000012	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10000013	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10000014	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10000015	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10000016	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.
10000017	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10000025	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10001002	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10001006	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.
10001007	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10001008	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.
10001010	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.
10002000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.
10002003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
10002005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.

20 rows × 298 columns

In [3]:
```
#将dataframe转换为纯数字矩阵
data_matrix = data.as_matrix()
col = data.shape[1]
row = data.shape[0]

for size in [0.01, 0.05, 0.1, 0.2, 0.3, 0.5]:
    hash_size = int(col * size)
    lsh = LSHash(hash_size, input_dim = row )
    #将矩阵导入，生成哈希表
    for  col_index in range(col):
        #将矩阵的每一列导入哈希表，用vipno的值作为extra_data
        lsh.index(data_matrix[:, col_index], extra_data = data.columns[col_index])
    else:
        #随机取列
        vipno_pos = 14
        for k in [1, 2, 3, 4, 5] :
            #再用hash_table的查询结果
            hash_table =  lsh.query(data_matrix[:, vipno_pos], num_results= k + 1, distance_func="euclidean")
            print('hash_size:' + str(hash_size) +
                  ', k:' + str(k) +
                  ', vipno:' + str(data.columns[vipno_pos]) +
                  ', result:')
            result = []
            for res in hash_table:
                result.append(res[0][1])
            else:
                print(result[1:])
```

```
hash_size:2, k:1, vipno:1590142151757, result:
[2900000549289]
hash_size:2, k:2, vipno:1590142151757, result:
[2900000549289, 1591011326672]
hash_size:2, k:3, vipno:1590142151757, result:
[2900000549289, 1591011326672, 1593140967467]
hash_size:2, k:4, vipno:1590142151757, result:
[2900000549289, 1591011326672, 1593140967467, 1590151470542]
hash_size:2, k:5, vipno:1590142151757, result:
[2900000549289, 1591011326672, 1593140967467, 1590151470542, 1592140505983]
hash_size:14, k:1, vipno:1590142151757, result:
[]
hash_size:14, k:2, vipno:1590142151757, result:
[]
hash_size:14, k:3, vipno:1590142151757, result:
[]
hash_size:14, k:4, vipno:1590142151757, result:
[]
hash_size:14, k:5, vipno:1590142151757, result:
[]
hash_size:29, k:1, vipno:1590142151757, result:
[]
hash_size:29, k:2, vipno:1590142151757, result:
[]
hash_size:29, k:3, vipno:1590142151757, result:
[]
hash_size:29, k:4, vipno:1590142151757, result:
[]
hash_size:29, k:5, vipno:1590142151757, result:
[]
hash_size:59, k:1, vipno:1590142151757, result:
[]
hash_size:59, k:2, vipno:1590142151757, result:
[]
hash_size:59, k:3, vipno:1590142151757, result:
[]
hash_size:59, k:4, vipno:1590142151757, result:
[]
hash_size:59, k:5, vipno:1590142151757, result:
[]
hash_size:89, k:1, vipno:1590142151757, result:
[]
hash_size:89, k:2, vipno:1590142151757, result:
[]
hash_size:89, k:3, vipno:1590142151757, result:
[]
hash_size:89, k:4, vipno:1590142151757, result:
[]
hash_size:89, k:5, vipno:1590142151757, result:
[]
hash_size:149, k:1, vipno:1590142151757, result:
[]
hash_size:149, k:2, vipno:1590142151757, result:
[]
hash_size:149, k:3, vipno:1590142151757, result:
[]
hash_size:149, k:4, vipno:1590142151757, result:
[]
hash_size:149, k:5, vipno:1590142151757, result:
[]
```

In [ ]:

二、讨论分析

Lshash 是一个实现局部敏感哈希的 python 包它的功能包括：

1. 通过使用 NumPy 数组的计算将高维数据快速哈希。
2. 支持多哈希索引
3. 内置通常的距离函数/排名输出

作业一要求通过对不同客户 vipno 进行哈希，通过比较 vipno 所购买 pluno 的情况，比较不同 vipno 的相似性，并通过改变 hash_size 和 k 两个参数，观察其对结果的影响。

hash_size 是要生成的哈希表的长度，k 值是对某个特定的输入，查找出和它相似的结果的数量。

从结果中可以看到仅当 hash_size = 2 时， k 取 1~5 都能够查询出结果；而当 hash_size 取其他更大的值时，除了输入之外不能查询出别的结果。说明所给的数据，只能分出两组特征不同的簇，而对于 k 取 1~5 都能查询出结果，并不能看出两个簇分配的是否均匀，所以不妨将 k 取更大的值，并输出 result 的长度：

```
hash_size:2, k:200, vipno:1590142151757, result:
[1590142156790, 1590142194532, 1595132332932, 1590142192491, 1590142202978, 15901513002
69, 1591140691788, 2900000175648, 1590141216914, 1595151575662, 1590151784960, 29000002
24278, 2900001118613, 1595150991142, 1590151207971, 1595151786686, 1591150401421, 15910
16151613, 1593160596081, 2900000908079, 2900001333566, 1590142434362, 1595150512170, 29
00001465816, 2900002934007, 2900000765085, 1592131140216, 1595132684185, 2900001241540,
2900001229272, 1598140023120, 1590142517867, 1595130381604, 2900002934298, 159514243218
9, 1591016150630, 1596160013077, 1590142190480, 1592015013285, 2900003114613, 159516019
4618, 2900000849273, 1591014756582, 2900000575899, 1590142519632, 2900000194298, 290000
1467100, 2900003115917, 2900002934236, 2900003109510, 1591016439575, 1591015457273, 159
1015088262, 1595142065844, 2900002936674, 2900000903944, 1591040462983, 2900001432436,
1591014932061, 1592014001436, 1591030270772, 1590151103907, 1591013368588, 159101545254
4, 1590142242059, 1591015592073, 2900003116167, 1591040084970, 1591013884927, 290000293
4557, 2900000667914, 2900001538596, 1591015077624, 1598140109237, 1590151534688, 290000
3113500, 18033305699, 2900001452236, 1590151101026, 2900000330863]
80
hash_size:2, k:100, vipno:1590142151757, result:
[1590142156790, 1590142194532, 1595132332932, 1590142192491, 1590142202978, 15901513002
69, 1591140691788, 2900000175648, 1590141216914, 1595151575662, 1590151784960, 29000002
24278, 2900001118613, 1595150991142, 1590151207971, 1595151786686, 1591150401421, 15910
16151613, 1593160596081, 2900000908079, 2900001333566, 1590142434362, 1595150512170, 29
00001465816, 2900002934007, 2900000765085, 1592131140216, 1595132684185, 2900001241540,
2900001229272, 1598140023120, 1590142517867, 1595130381604, 2900002934298, 159514243218
9, 1591016150630, 1596160013077, 1590142190480, 1592015013285, 2900003114613, 159516019
4618, 2900000849273, 1591014756582, 2900000575899, 1590142519632, 2900000194298, 290000
1467100, 2900003115917, 2900002934236, 2900003109510, 1591016439575, 1591015457273, 159
1015088262, 1595142065844, 2900002936674, 2900000903944, 1591040462983, 2900001432436,
1591014932061, 1592014001436, 1591030270772, 1590151103907, 1591013368588, 159101545254
4, 1590142242059, 1591015592073, 2900003116167, 1591040084970, 1591013884927, 290000293
4557, 2900000667914, 2900001538596, 1591015077624, 1598140109237, 1590151534688, 290000
3113500, 18033305699, 2900001452236, 1590151101026, 2900000330863]
80
```

可以看到当 k=100,200 时，其结果的长度都为 80，多次运行之后结果的长度都不会超过总的 vipno 的数目 298 的一半，也就是说明两个簇的分配是均匀的。

综上所述，得到结论：

1. 通过购买 pluno 的情况，只能获得两组具有不同特征的散列，所给数据特征数不充分
2. LSHash 得到的结果是均匀的