Q1（数据预处理 + knn）

姓名：王依睿

学号：1552651

a) 获得矩阵

1. 使用`pandas.read_csv`读入数据

	uid	sldat	pno	cno	vipno	id	pluno	bcd	pluname	spec	...	dptno	dptname	bndn
1364	16072311561807577	2016-07-23 11:56:37	18.0	8338.0	13205496418	6.0	10000006	6933266600025	百事可乐600ml	1*24	...	10000	可乐	10007.
1362	16060814461906749	2016-06-08 14:46:05	19.0	8306.0	13205496418	2.0	10110017	6956416200067	美汁源果粒橙420ml	1*24	...	10110	橙汁	10080.
1363	16072311561807577	2016-07-23 11:56:37	18.0	8338.0	13205496418	3.0	10110017	6956416200067	美汁源果粒橙420ml	1*24	...	10110	橙汁	10080.
1291	16051216561827784	2016-05-12 16:56:01	18.0	8306.0	13205496418	2.0	10119019	6938166929689	和丝露蓝莓汁828ml	1*6	...	10119	其他果汁饮料	10719.
1290	16051216561827784	2016-05-12 16:56:01	18.0	8306.0	13205496418	3.0	10119020	6938166929320	和丝露芒果汁828ml	1*6	...	10119	其他果汁饮料	10719.
		2016-												

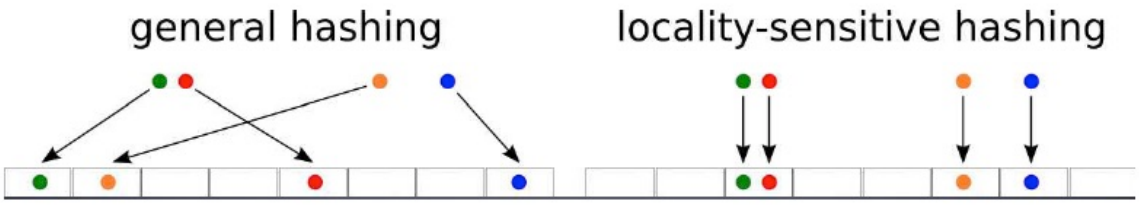
2. 将trade文件中的记录按照vipno分组，汇总每个vipno购买pluno的总记录数；将以上汇总的vipno分组转为一个矩阵，其中**矩阵行**代表为pluno，**矩阵列**代表为vipno，**矩阵元素**为该vipno所购买pbaoluno的总金额（取整，四舍五入），使用`DataFrame`格式存储

vipno	13205496418	15954611837	18033305699	18678696982	1590140040664	1590140118226	1590140305015	1590140400093	1590140606433	159014061131
pluno										
10000006	2	0	0	0	0	0	0	0	0	
10110017	4	0	0	0	0	0	0	0	0	
10119019	12	0	0	0	0	0	0	0	0	
10119020	12	0	0	0	0	0	0	0	0	
10130001	2	0	0	0	0	0	0	0	0	
10132002	3	0	0	0	0	0	0	0	0	
10136005	22	0	0	0	0	0	0	0	0	
10201002	16	0	0	0	0	0	0	0	0	
10300019	12	0	0	0	0	0	0	0	0	
10401008	1	0	0	0	0	0	0	0	0	
11020014	28	0	0	0	0	0	0	0	0	
11220073	18	0	0	0	0	0	0	0	0	
11300123	12	0	0	0	0	0	0	0	0	
11511084	78	0	0	0	0	0	0	0	0	
14020005	13	0	0	0	0	0	0	0	0	
14078003	6	0	0	0	0	0	0	0	0	
14091006	5	0	0	0	0	0	0	0	0	
14091009	3	0	3	0	0	0	0	0	0	
14091047	10	0	0	0	0	0	0	0	0	
14092013	3	0	0	0	0	0	0	0	0	
14092014	3	0	0	0	0	0	0	0	0	
14092015	6	0	0	0	0	0	0	0	0	
14092017	3	0	0	0	0	0	0	0	0	
14092068	6	0	0	0	0	0	0	0	0	

b) knn

1. 安装并引入`lshash3`包

lsh原理：



一个哈希函数族满足如下条件时，被称为是(R, cR, P_1, P_2)-sensitvie，对于任意两个点 p , $q \in \mathbb{R}^d$:

- if $\|p-q\| \leq R$ then $\Pr_{\mathcal{H}}[h(q)=h(p)] \geq P_1$,
- if $\|p-q\| \geq cR$ then $\Pr_{\mathcal{H}}[h(q)=h(p)] \leq P_2$.

为了让局部敏感哈希函数族起作用，需要满足$c > 1$，$P_1 > P_2$.

2. 将每条`vip`进行哈希操作并记录
3. 随机取一个点`vipno`
4. 利用`lsh`计算周围点到自己的距离

```
# 利用lsh计算周围点到自己的距离
def lsh(hash_size, input_dim, query_point):
    global data;
    lsh = LSHash(hash_size, input_dim);
    for vipno in data:
        lsh.index(list(data[vipno]), extra_data=vipno)
    return lsh.query(query_point)
```

5. 选出离自己最近的k个点

```
# 选出离自己最近的k个点
def knn(k, hash_size, input_dim, query_point):
    k_query = lsh(hash_size, input_dim, query_point)[1: k + 1];
    k_vipno = [];
    k_distance = [];
    for item in k_query:
        k_vipno.append(item[0][1]);
        k_distance.append(item[1]);
    knn_df = DataFrame({'vipno':k_vipno, 'distance': k_distance})
    return knn_df
```

6. 将每个矩阵列通过Pythong lshash进行索引处理，hash_size为全体vipno(非重复)总数的0.01, 0.05, 0.1, 0.2, 0.3, 0.5; 任意选择一个vipno, 然后输出该vipno对应knn的输出vipno (k = 1, 2, 3, 4, 5)

```
# 选一个vipno对不同的hash_size和k进行计算
def vip_knn(input_dim, vipno):
    print("vipno: ", vipno);
    query_point = data[vipno];
    global hash_size_list;
    global k_list;
    vip_knn_df = DataFrame(columns = k_list, index = hash_size_list);
    for hash_size in hash_size_list:
        for k in k_list:
            knn_df = knn(k, hash_size, input_dim, query_point);
            print("hash_size: ", hash_size);
            print("k: ", k);
            print (knn_df);
```

结果

```
vipno:  1595151630699
hash_size:  3
k:  1
   distance          vipno
0     14899  1590151544861
hash_size:  3
k:  2
   distance          vipno
0     15146  1590151207971
1     15186  1590142192491
hash_size:  3
k:  3
   distance          vipno
0     14704  1595151575662
1     15155  2900000549289
2     15369  1595151110818
k:  4
   distance          vipno
0     15408  1590142516563
1     15423  1595151630507
2     15562  1592140505983
3     15726  1591140691788
hash_size:  3
k:  5
   distance          vipno
0     15186  1590142192491
1     15889  1592140611301
2     15998  2900002517941
3     16191  1595141299820
4     16432  1591015442491
```

```
hash_size:  15
k:  5
Empty DataFrame
Columns: [distance, vipno]
Index: []
hash_size:  30
k:  1
Empty DataFrame
Columns: [distance, vipno]
Index: []
hash_size:  30
k:  2
Empty DataFrame
Columns: [distance, vipno]
Index: []
hash_size:  30
k:  3
Empty DataFrame
Columns: [distance, vipno]
```

讨论

当`hash_size`为全体vipno(非重复)总数的0.01时能求得随机点的临近点，其余情况由于`hash_size`太大使得点在哈希表上太过稀疏，从而无法求得临近点。