Q2（k-means）

姓名：王依睿

学号：1552651

a) k-means

1. 引入`sklearn`和`KMeans`

```python
import sklearn
from sklearn.cluster import KMeans
```

2. k-means聚类并计算其silhouette系数

```python
# k-means聚类并计算其silhouette系数
def kmeans(n_clusters):
    X = StandardScaler().fit_transform(data.T)
    # Incorrect number of clusters
    labels = KMeans(n_clusters = n_clusters).fit_predict(X)
    silhouette_score = metrics.silhouette_score(X, labels, metric='euclidean')
    return silhouette_score
```

3. 选择不同的k进行k-means聚类并计算其silhouette系数

```python
# 选择不同的k进行k-means聚类并计算其silhouette系数
def vip_kmeans():
    n_clusters_list = pd.Series(range(2, int(math.sqrt(vipno_total / 2) + 40)))
    vip_kmeans_df = pd.DataFrame(columns=['silhouette_score'], index = n_clusters_list)
    for n_clusters in n_clusters_list:
        silhouette_score = kmeans(n_clusters)
        vip_kmeans_df['silhouette_score'][n_clusters] = silhouette_score
    return vip_kmeans_df
```

结果

标准化后

silhouette_score

2	0.48722
3	0.258304
4	0.409811
5	-0.0225924
6	-0.164264
7	0.341519
8	0.109789
9	0.184277
10	0.201069
11	-0.154637
12	0.279303
13	0.171036
14	-0.281389
15	-0.149187
16	0.256569
17	0.187516
18	-0.0967663

19	0.0399323
20	0.190137
21	0.238392
22	0.253588
23	0.276265
24	-0.14351
25	-0.019352
26	0.0447384
27	0.23375
28	-0.17327
29	0.121717
30	-0.2243
31	0.13538
32	0.0375936
33	-0.271416
34	-0.150653
35	0.18909
36	0.162469

36	0.162469
37	0.124659
38	0.189629
39	0.107688
40	0.167317
41	0.0933236
42	-0.0477688
43	-0.152667
44	-0.19394
45	0.195354
46	-0.0799995
47	-0.170349
48	-0.158859
49	-0.0380165
50	0.0929955

标准化前

	silhouette_score
2	0.941759

3	0.864978
4	0.806563
5	0.73279
6	0.302269
7	0.582366
8	0.295611
9	0.503593
10	0.264648
11	0.320026
12	0.108558
13	0.314193
14	0.221744
15	0.325281
16	0.284054
17	0.24598
18	0.312843
19	0.0630744

20	0.263849
21	0.163162
22	0.305543
23	0.21772
24	0.242047
25	0.260464
26	0.252603
27	0.10152
28	0.306447
29	0.12048
30	0.0659082
31	0.128939
32	-0.148261
33	0.0438414
34	0.0896596
35	0.0911414
36	0.0965219
37	0.18304

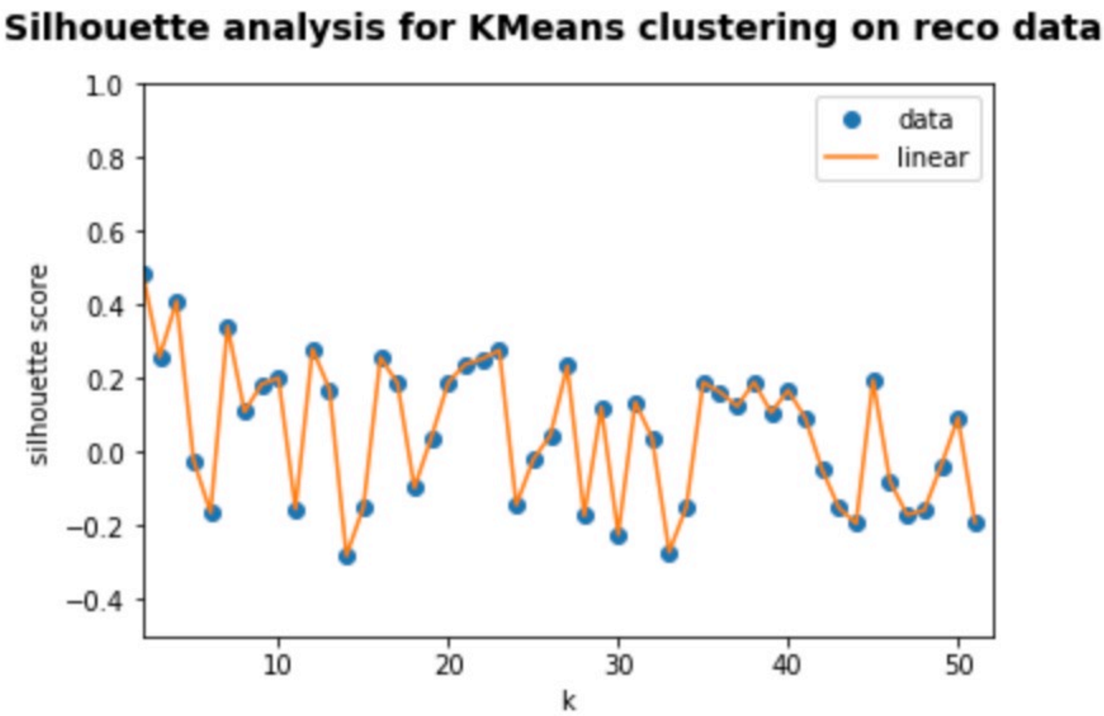
38	0.0655652
39	0.0567768
40	0.276519
41	0.0947079
42	0.0679171
43	0.134885
44	0.132513
45	0.0357655
46	0.116735
47	0.118276
48	0.106903
49	0.0428625
50	0.0765129

1. 利用`matplotlib.pyplot`以k值为横轴、Silhouette系数值为y轴，画出Silhouette系数值-k值的函数图

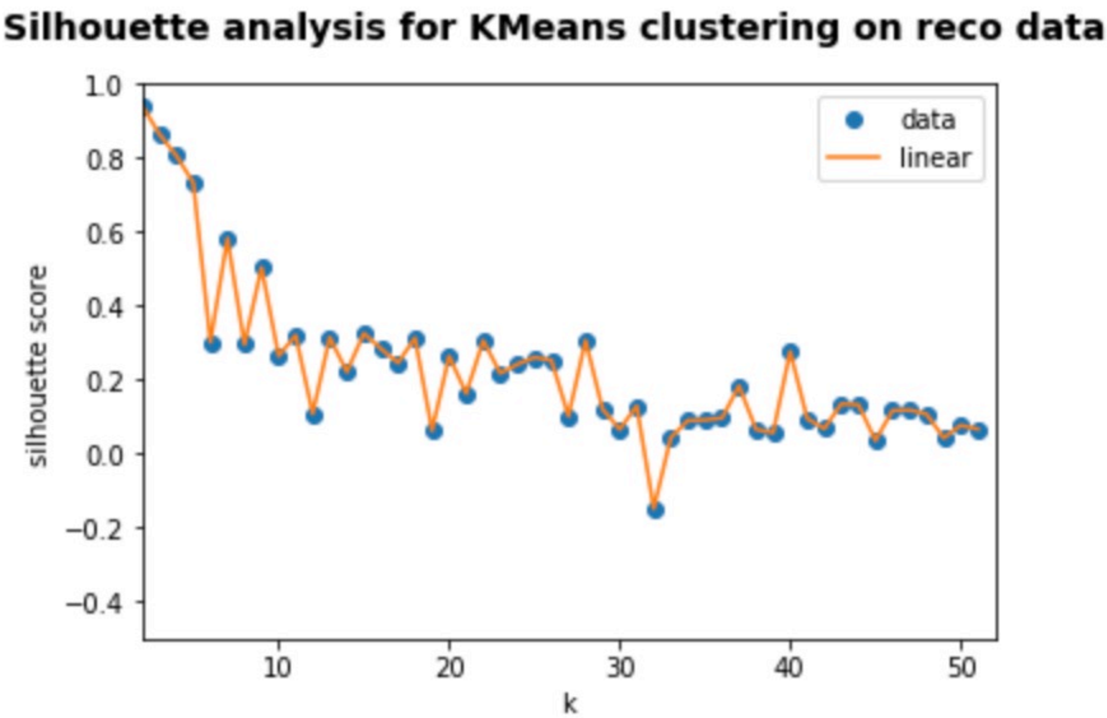
```
# 作图
def kmeans_plot(vip_kmeans_df):
    f = interp1d(vip_kmeans_df.index, vip_kmeans_df.silhouette_score)
    xnew = vip_kmeans_df.index.copy()
    plt.plot(vip_kmeans_df.index, vip_kmeans_df.silhouette_score, 'o',
xnew, f(xnew), '-')
    plt.xlim([2, int(math.sqrt(vipno_total / 2) + 40)])
```

```
    plt.ylim([-0.5, 1])
    plt.legend(['data', 'linear'], loc='best')
    plt.xlabel('k')
    plt.ylabel('silhouette score')
    plt.suptitle(
        "Silhouette analysis for KMeans clustering on reco data",
        fontsize=14, fontweight='bold')
    plt.show()
```

标准化后



标准化前



b) 验证lsh的knn结果

1. 选择聚类效果较好的 k = 2， 3，4时，查看聚类结果

```
X = StandardScaler().fit_transform(data.T)
kmeans_labels = KMeans(n_clusters = 2).fit_predict(X)
kmeans_labels
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)
```

2. 查看knn结果

```
hash_size:  3
k:  5
   distance          vipno
0     11056  2900000549289
1     11176  1593140967467
2     11466  1595142205462
3     11766  1590141216914
4     11790  1592140611301
```

结论

1. 若将Silhouette系数作为评价k-means聚类质量的标准，则k-means算法倾向于将数据点聚类成较少的类别。
2. 根据聚类结果来看，k-means聚类倾向于将极大多数点分为一簇，其现实意义可能是由于大多数人的购物习惯都较为相似，数据量较少也是其中一个原因。
3. 正因为极大多数点划分为同一簇，从而使得验证lsh的knn结果时，lsh的knn查询结果与输入vipno都在同一个簇。