聚类大作业

1. 数据介绍：

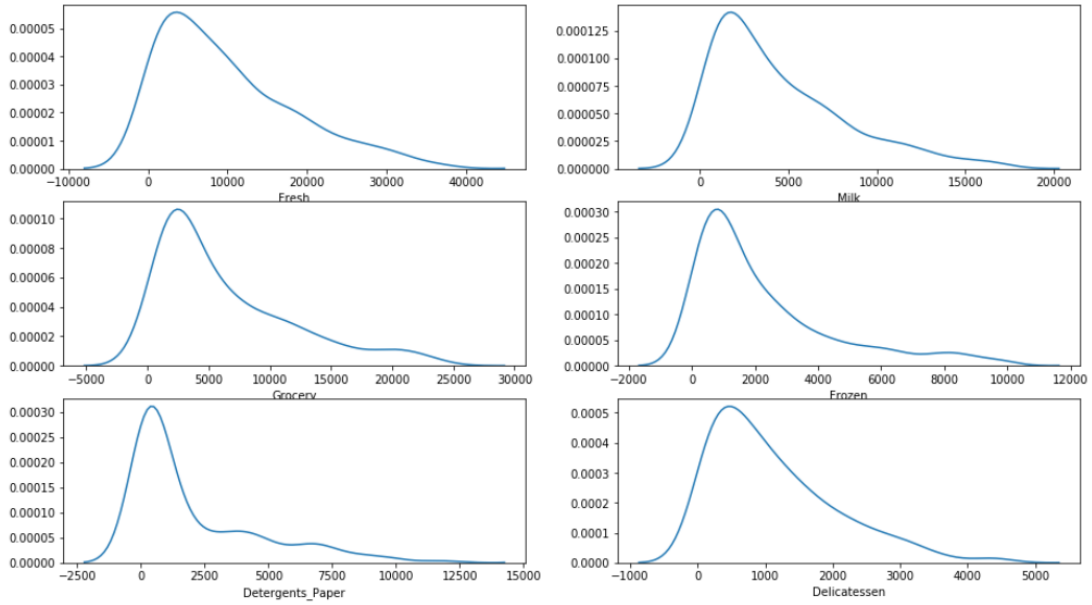
   本实验采用的数据集(data.xlsx)有七维数据，其中包含了用户 ID 和用户在 6 种不同商品上的消费额，总共有 440 条数据。如图：

id	Fresh	Milk	Grocery	Frozen	Detergents_	Delicatessen
1	12669	9656	7561	214	2674	1338
2	7057	9810	9568	1762	3293	1776
3	6353	8808	7684	2405	3516	7844
4	13265	1196	4221	6404	507	1788
5	22615	5410	7198	3915	1777	5185
6	9413	8259	5126	666	1795	1451
7	12126	3199	6975	480	3140	545
8	7579	4956	9426	1669	3321	2566
9	5963	3648	6192	425	1716	750
10	6006	11093	18881	1159	7425	2098
11	3366	5403	12974	4400	5977	1744
12	13146	1124	4523	1420	549	497
13	31714	12319	11757	287	3881	2931
14	21217	6208	14982	3095	6707	602
15	24653	9465	12091	294	5058	2168
16	10253	1114	3821	397	964	412
17	1020	8816	12121	134	4508	1080
18	5876	6157	2933	839	370	4478
19	18601	6327	10099	2205	2767	3181
20	7780	2495	9464	669	2518	501
21	17546	4519	4602	1066	2259	2124
22	5567	871	2010	3383	375	569
23	31276	1917	4469	9408	2381	4334
24	26373	36423	22019	5154	4337	16523
25	22647	9776	13792	2915	4482	5778
26	16165	4230	7595	201	4003	57
27	9898	961	2861	3151	242	833
28	14276	803	3045	485	100	518
29	4113	20484	25957	1158	8604	5206
30	43088	2100	2609	1200	1107	823
31	18815	3610	11107	1148	2134	2963
32	2612	4339	3133	2088	820	985
33	21632	1318	2886	266	918	405
34	29729	4786	7326	6130	361	1083
35	1502	1979	2262	425	483	395
36	688	5491	11091	833	4239	436
37	29955	4362	5428	1729	862	4626
38	15168	10556	12477	1920	6506	714
39	4591	15729	16709	33	6956	433

2. 数据分析：

将六维数据分别绘图，利用 np.percentile 方法找到大于 95%数据的点，并在结果中将其隐藏，结果如下：

```
# 显示数据，去除大于95%的极端值
plt.figure(figsize=(16, 9))
for i, col in enumerate(list(data.columns)[1:]):
    plt.subplot(321 + i)
    q95 = np.percentile(data[col], 95)
    sns.distplot(data[data[col] < q95][col], hist=False)
plt.show()
```



可以看到不同商品的年消费基本符合大于 0 的正态分布，分布情况也比较类似。

3. 数据预处理

a) 首先去除数据的极值，减少离群点的影响。

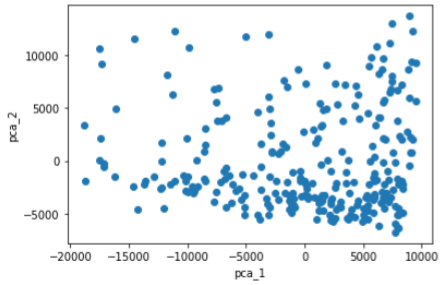
```
features = data[['Fresh', 'Milk', 'Grocery', 'Frozen', 'Detergents_Paper', 'Delicatessen']]

ids = []
for i in list(features.columns):
    left = np.percentile(features[i], 25) # 前25%
    right = np.percentile(features[i], 75) # 后25%
    interval = 1.6 * (right - left) / 2
    low = left - interval
    high = right + interval
    ids.extend(list(features[(features[i] <= low) | (features[i] >= high)].index))
ids = list(set(ids))
features = features.drop(ids)
```

首先获取 1/4 点和 3/4 点 left 和 right。然后计算出 interval = 1.6*(right - left) / 2。再根据 interval 的值决定极值点的范围即 low = left - interval，high = right + interval。获取离群点的序号，并在数据中去除它们。

b) 然后使用 PCA 降维，将数据降至两维，在两维上分布如下：

PCA降维，降到两维

```
# 每一列均值
data_mean = np.mean(features, axis=0)

# 均值归一
features = features - data_mean

# 协方差
cov = np.cov(features.transpose())

# 特征值和特征向量
eigVals, eigVectors = np.linalg.eig(cov)

# 选择前两个特征向量
pca_mat = eigVectors[:, :2]
pca_data = np.dot(features, pca_mat)
pca_data = pd.DataFrame(pca_data, columns=['pca1', 'pca2'])

# 两个主成分的散点图
plt.subplot(111)
plt.scatter(pca_data['pca1'], pca_data['pca2'])
plt.xlabel('pca_1')
plt.ylabel('pca_2')
plt.show()
```



关于 PCA：

PCA，即主成分分析，是一种对多维数据进行降维的方法。在本次实验中，需要对含有多个变量的数据进行观测，因为许多变量之间可能存在相关性，如果分别对单个指标进行分析，这种分析往往是孤立的，可能会丢失很多有用的信息。而 PCA 可以在减少所需要分析指标的同时，尽量减少原指标包含信息的损失，从而达到对收集数据全面分析的目的。PCA 的主要思想是将 n 维特征映射到 k 维上，这 k 维是全新的正交特征也被称为主成分，是在原有 n 维特征的基础上重新构造出来的 k 维特征。PCA 的工作就是从原始的空间中顺序地找一组相互正交的坐标轴，新的坐标轴的选择与数据本身是密切相关的。其中，第一个新坐标轴选择是

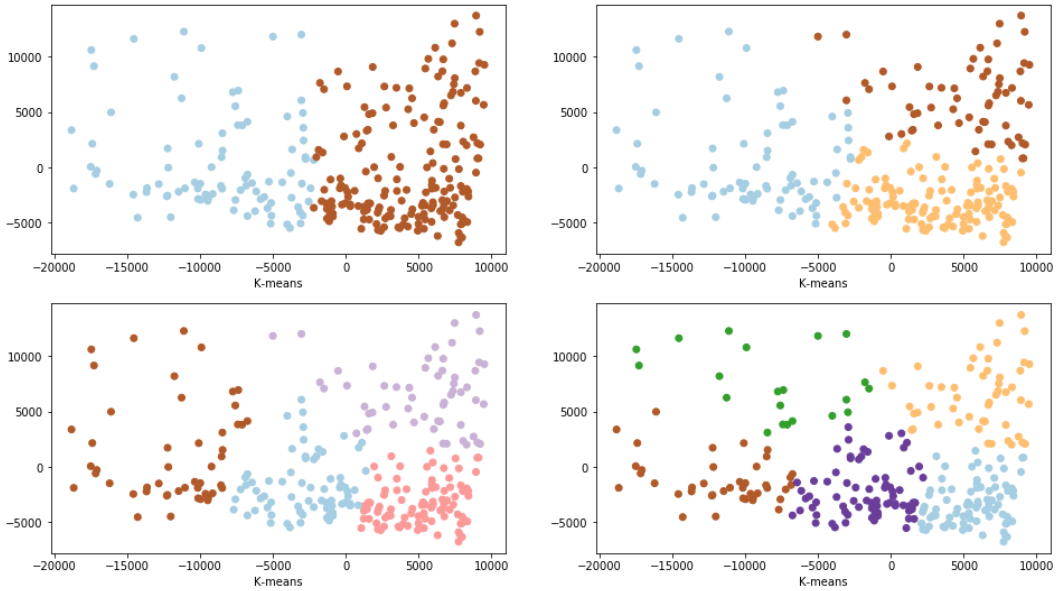
原始数据中方差最大的方向，第二个新坐标轴选取是与第一个坐标轴正交的平面中使得方差最大的，第三个轴是与第 1,2 个轴正交的平面中方差最大的。依次类推，可以得到 n 个这样的坐标轴。通过这种方式获得的新的坐标轴，我们发现，大部分方差都包含在前面 k 个坐标轴中，后面的坐标轴所含的方差几乎为 0。于是，我们可以忽略余下的坐标轴，只保留前面 k 个含有绝大部分方差的坐标轴。事实上，这相当于只保留包含绝大部分方差的维度特征，而忽略包含方差几乎为 0 的特征维度，实现对数据特征的降维处理。

4. 聚类处理

在本次实验中，我们选择的聚类算法是 k-means。

选择 k-means 的原因，是因为它是目前使用得最广泛的聚类算法，同时计算复杂度低。

```
random_state = 80 # 随机数种子
n_cluster = np.arange(2, 6) # 聚簇为2 ~ 5
plt.figure(figsize=(16, 9))
for i, k in zip([0, 1, 2, 3], n_cluster):
    kmeans = KMeans(n_clusters=k, random_state=random_state)
    cluster1 = kmeans.fit_predict(pca_data)

    # 绘图
    plt.subplot(221 + i)
    plt.scatter(pca_data['pca1'], pca_data['pca2'], c=cluster1, cmap=plt.cm.Paired)
    plt.xlabel('K-means')
plt.show()
```



在聚类处理中，首先设定随机数种子，初始化分类器并对降维后的 PCA_DATA 进行聚类。尝试 k 值为 2、3、4、5。之后使用 matplotlib 库将聚类结果显示出来。

5. 聚类结果分析

在本次实验中，我们采用了轮廓系数来对聚类的效果进行评估。

轮廓系数(silhouette coefficient) 结合了凝聚度和分离度，其计算步骤如下：

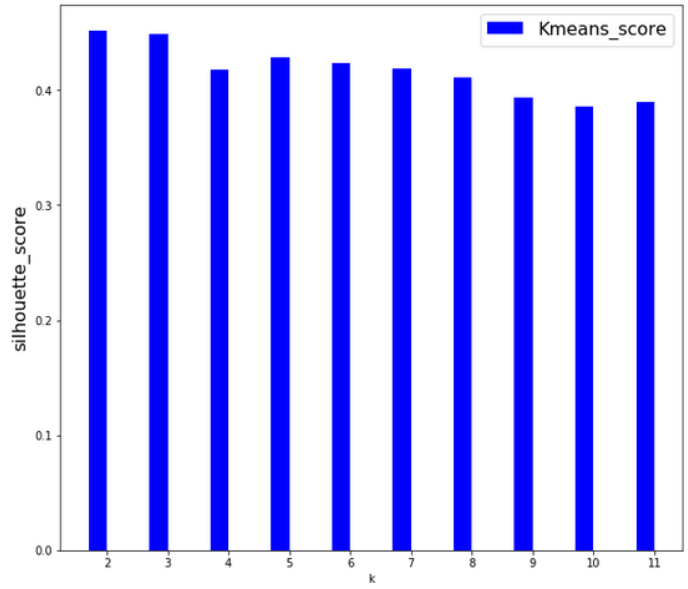
i. 对于第 i 个对象，计算它到所属簇中所有其他对象的平均距离，记 ai （体现凝聚度）

ii. 对于第 i 个对象和不包含该对象的任意簇，计算该对象到给定簇中所有对象的平均距离，记 bi （体现分离度）

iii. 第 i 个对象的轮廓系数为 si = (bi-ai)/max(ai, bi)

本次实验调用了 sklearn 库的 silhouette_score 函数来计算轮廓系数的平均值，轮廓系数越接近 1 说明聚类效果越好，对 k 从 2~11 变化时，轮廓系数的值进行绘制。

```
score = [] # 存储结果
random_state = 80 # 种子
n_cluster = np.arange(2, 12) # 2~11
for k in n_cluster:
    kmeans = KMeans(n_clusters=k, random_state=random_state)
    cluster = kmeans.fit_predict(pca_data)
    score.append(silhouette_score(pca_data, cluster))

# 绘图
sil_score = pd.DataFrame({'k': np.arange(2, 12), 'score_kmean': score})
plt.figure(figsize=(10, 9))
plt.bar(sil_score['k'] - 0.15, sil_score['score_kmean'], width=0.3, facecolor='blue', label='Kmeans_score')
plt.xticks(np.arange(2, 12))
plt.legend(fontsize=16)
plt.ylabel('silhouette_score', fontsize=16)
plt.xlabel('k')
plt.show()
```



可以看到轮廓系数在 0.4~0.5 之间，k-means 在聚类类别为 2~3 时效果较好，聚类数增多时，轮廓系数平滑的降低，但是幅度并不大，说明 k-means 算法在本数据集上表现比较稳定。

6. 总结

本次实验学习了用 python 进行数据预处理、数据聚类的基本方法，学会了常见库的调用方式和结果的表示方式。掌握了使用 k-means 对数据进行聚类的过程。同时，在此过程中也对 k-means 聚类方法更为熟悉，了解了 k-means 的

优点与不足。对于优点，k-means 使用简单，计算的复杂度低，思想也比较简单，易于理解。而对于缺点，k-means 分类的结果依赖于分类中心的初始化，有进入局部最优而不是全局最优的可能性，对噪声敏感，并且只适用于连续数据。