**实验11：Python聚类分析**

**一：实验目的和要求**

1. 掌握利用Python实现K-Means聚类分析的方法
2. 掌握利用Python实现系统聚类分析的方法

**二：内容和方法**

分别使用K-means和系统聚类分析方法实现对手写数字识别。步骤如下：

1. 导入digits数据：

from sklearn.datasets import load\_digits

digits = load\_digits()

1. 标准化数据和PCA降维。（可选步骤）
2. 聚类，并想办法获取每个类代表的是什么数字（标签）。
3. 利用聚类结果，对每个样本预测其类别，并根据真实的类别计算错误率。

**from** sklearn.datasets **import** load\_digits  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.preprocessing **import** StandardScaler  
**from** sklearn.decomposition **import** PCA  
**from** sklearn.cluster **import** KMeans  
**from** sklearn.cluster **import** AgglomerativeClustering  
**import** numpy **as** np  
**from** munkres **import** Munkres  
**import** pandas **as** pd  
*#提取数据*digits=load\_digits()  
data=digits[**'data'**]  
data.shape *#1797个图片数据，每张图片为8\*8=64个像素值*target=digits[**'target'**]  
*#print(data,target,digits,data.shape,digits['images'],digits['images'].shape)  
#标准化数据*SS=StandardScaler().fit(data)  
data\_SS=SS.transform(data)  
*#pca降维*pca=PCA(n\_components=8).fit(data\_SS)  
data\_pca=pca.transform(data\_SS)  
*#K-Means聚类*y\_pred=KMeans(n\_clusters=10).fit\_predict(data\_pca)  
print(target,y\_pred)  
*#系统聚类*y1\_pred=AgglomerativeClustering(affinity=**'euclidean'**,linkage=**'ward'**,n\_clusters=10).fit\_predict(data\_pca)  
print(y1\_pred)  
*#聚类结果标签映射真实标签***def** best\_map(L1,L2):  
Label1 = np.unique(L1) *# 去除重复的元素，由小大大排列* nClass1 = len(Label1) *# 标签的大小* Label2 = np.unique(L2)  
 nClass2 = len(Label2)  
 nClass = np.maximum(nClass1,nClass2)  
 G = np.zeros((nClass,nClass))  
 **for** i **in** range(nClass1):  
 ind\_cla1 = L1 == Label1[i]  
 ind\_cla1 = ind\_cla1.astype(float)  
 **for** j **in** range(nClass2):  
 ind\_cla2 = L2 == Label2[j]  
 ind\_cla2 = ind\_cla2.astype(float)  
 G[i,j] = np.sum(ind\_cla2 \* ind\_cla1)  
 m = Munkres()  
 index = m.compute(-G.T)  
 index = np.array(index)  
 c = index[:,1]  
 newL2 = np.zeros(L2.shape)  
 **for** i **in** range(nClass2):  
 newL2[L2 == Label2[i]] = Label1[c[i]]  
 **return** newL2  
**def** err\_rate(gt\_s,s):  
 c\_x=best\_map(gt\_s,s)  
 err\_x=np.sum(gt\_s[:]!=c\_x[:])  
 missrate=err\_x.astype(float)/(gt\_s.shape[0])  
 **return** missrate  
  
print(**'KMeans聚类后的预测标签：'**,best\_map(target,y\_pred),**'\n'**,**'KMeans聚类后预测样本的错误率'**,err\_rate(target,y\_pred))  
  
print(**'系统聚类后的预测标签：'**,best\_map(target,y1\_pred),**'\n'**,**'系统聚类后预测样本的错误率'**,err\_rate(target,y1\_pred))

