实验二　聚类 K-means 算法

【实验目的】

1. 了解 Anaconda 和 python 的使用与设置。
2. 掌握聚类 K-means 算法的基本原理和实现方法。

【实验学时】

建议 2 学时

【实验环境配置】

1、 Windows 环境
2、 Anaconda
3、 Pandas

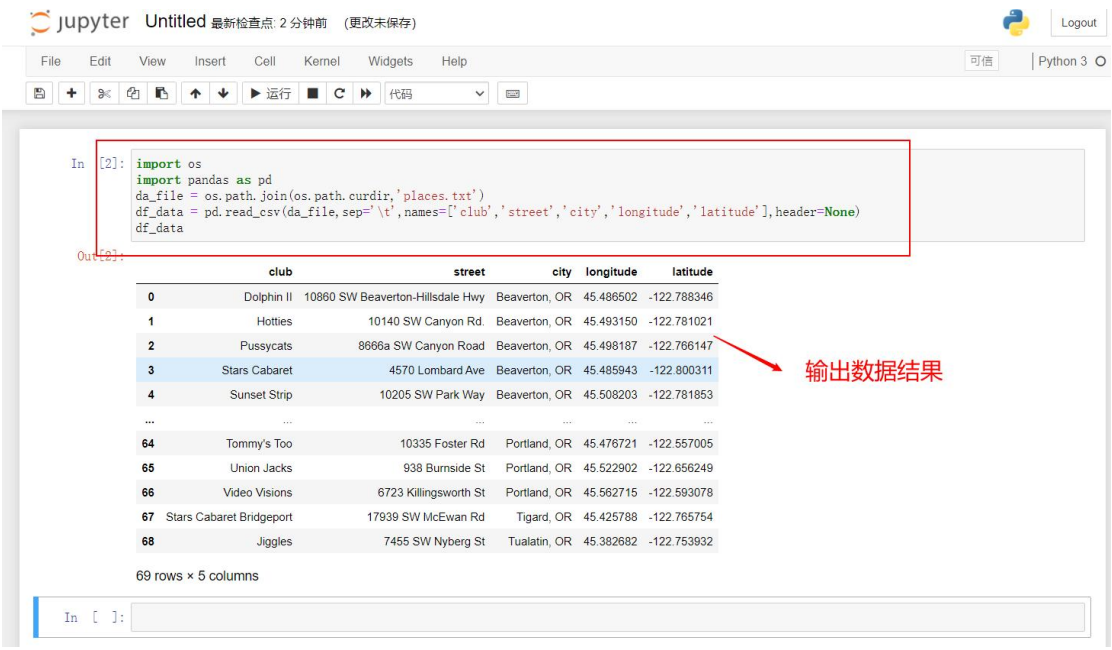
【实验原理】

数据挖掘中聚类 K-means 算法的实现：

1、Pandas 数据导入
2、K-means 算法
3、sklearn 库的 K-means 模块使用

【实验步骤】

1. 打开 Anaconda 的 jupyter notebook，创建实验二，明确标题和步骤。
2. 导入 pandas 库和 os 库，从 places.txt 数据文件中，导入实验数据。
   （注：可能涉及的函数 os.path.join, os.walk, pandas.read_csv）



	club	street	city	longitude	latitude
0	Dolphin II	10860 SW Beaverton-Hillsdale Hwy	Beaverton, OR	45.486502	-122.788346
1	Hotties	10140 SW Canyon Rd.	Beaverton, OR	45.493150	-122.781021
2	Pussycats	8666a SW Canyon Road	Beaverton, OR	45.498187	-122.766147
3	Stars Cabaret	4570 Lombard Ave	Beaverton, OR	45.485943	-122.800311
4	Sunset Strip	10205 SW Park Way	Beaverton, OR	45.508203	-122.781853
...	...	...	...	...	...
64	Tommy's Too	10335 Foster Rd	Portland, OR	45.476721	-122.557005
65	Union Jacks	938 Burnside St	Portland, OR	45.522902	-122.656249
66	Video Visions	6723 Killingsworth St	Portland, OR	45.562715	-122.593078
67	Stars Cabaret Bridgeport	17939 SW McEwan Rd	Tigard, OR	45.425788	-122.765754
68	Jiggles	7455 SW Nyberg St	Tualatin, OR	45.382682	-122.753932

69 rows × 5 columns

3. 对导入的实验数据，根据《机器学习实战》教程第十章，完成 K-means 算法，并完成对于 places.txt 数据的聚类

65	Union Jacks	938 Burnside St	Portland, OR	45.522902	-122.656249
66	Video Visions	6723 Killingsworth St	Portland, OR	45.562715	-122.593078
67	Stars Cabaret Bridgeport	17939 SW McEwan Rd	Tigard, OR	45.425788	-122.765754
68	Jiggles	7455 SW Nyberg St	Tualatin, OR	45.382682	-122.753932

69 rows × 5 columns

In [ ]:
```
import numpy as np
df_data['cluster'] = np.zeros(len(df_data),int)

def distEclud(vecA,vecB):
    return np.sqrt(np.power(vecA - vecB),2)
```

In [ ]:
```
def andCent(df_data, k):
    rand_sample = df_data.sample(n=k)
    centroids = rand_sample[['longitude','latitude','cluster']].copy(deep=True)
    centroids.loc[:,'cluster'] = [i for i in range(k)]
    return centroids
```

In [3]:
```
df_data.sample(n=3)
```

Out[3]:

	club	street	city	longitude	latitude
56	Safari Show Club	3000 SE Powell Blvd	Portland, OR	45.497091	-122.634581
44	Mr. Peeps	709 122nd Ave	Portland, OR	45.527863	-122.537726
28	G-Spot Airport	8654 Sandy Blvd	Portland, OR	45.554263	-122.574167

In [ ]:

```
# 简单k均值聚类算法
#       数据集   中心数量    距离算法             初始聚类中心算法
def kMeans(dataSet, k, distMeas=distEclud, createCent=randCent):
    m = shape(dataSet)[0]                # 样本个数
    clusterAssment = mat(zeros((m,2)))# 样本标记 分配结果 第一列索引 第二列误差
    centroids = createCent(dataSet, k)# 初始聚类中心
    clusterChanged = True# 设置质心是否仍然发送变化
    while clusterChanged:
        clusterChanged = False
        for i in range(m): #对每个样本 计算最近的中心
        # 更新 样本所属关系
            minDist = inf; minIndex = -1 # 距离变量 以及 最近的中心索引
            for j in range(k): # 对每个中心
                distJI = distMeas(centroids[j,:],dataSet[i,:])# 计算距离
                if distJI < minDist:
                    minDist = distJI; minIndex = j# 得到最近的 中心 索引
            if clusterAssment[i,0] != minIndex: clusterChanged = True
            # 所属索引发生了变化 即质心还在变化，还可以优化
            clusterAssment[i,:] = minIndex,minDist**2 # 保存 所属索引 以及距离平方 用以计算误差平方和 SSE
        # 更新质心
        print (centroids) # 每次迭代打印质心
        for cent in range(k):#
            ptsInClust = dataSet[nonzero(clusterAssment[:,0].A==cent)[0]]# 数组过滤 得到各个中心所属的样本
            centroids[cent,:] = mean(ptsInClust, axis=0) # 按列求平均 得到新的中心
    return centroids, clusterAssment# 返回质心 和各个样本分配结果
```

```
def kMeansTest(k=5):
    MyDatMat = mat(loadDataSet("testSet.txt"))
    MyCenters, ClustAssing = kMeans(MyDatMat, k)
```

聚类结果

```
[[-122.4201447    45.63777657]
 [-122.80084772   45.58768636]]
[[-122.53877368   45.52189292]
 [-122.68446173   45.50685998]]
[[-122.54432282   45.52042118]
 [-122.69133217   45.5067651 ]]
[[-122.54868607   45.51882187]
 [-122.69551477   45.50729503]]
sseSplit, and notSplit:  3043.2633161055337 0.0
the bestCentToSplit is:  0
the len of bestClustAss is:  69
[[-122.40274828   45.47330431]
 [-122.60008617   45.43137835]]
[[-122.4568086    45.4961344 ]
 [-122.56706156   45.52335936]]
sseSplit, and notSplit:  505.61960820164956 2191.824427523823
[[-122.83706043   45.5352803 ]
 [-122.65928613   45.56543235]]
[[-122.77486818   45.48294173]
 [-122.66434021   45.51686239]]
[[-122.7685635    45.48143208]
 [-122.66304867   45.51878967]]
sseSplit, and notSplit:  1486.777796110165 851.4388885817106
the bestCentToSplit is:  1
the len of bestClustAss is:  39
[[-122.60121906   45.55275915]
 [-122.45864022   45.54073252]]
[[-122.57664775   45.52983775]
 [-122.4927627    45.4967901 ]]
[[-122.57768158   45.53263337]
 [-122.49860291   45.49496564]]
sseSplit, and notSplit:  464.7205983452951 1486.777796110165
[[-122.76291259   45.45669646]
 [-122.77510808   45.59515597]]
[[-122.761804     45.46639582]
 [-122.842918     45.646831  ]]
sseSplit, and notSplit:  234.49990074938086 1698.1650028283193
[[-122.66873813   45.49483676]
 [-122.70574529   45.4891937 ]]
[[-122.65515417   45.51142475]
 [-122.72620467   45.577709  ]]
[[-122.65238871   45.5011109 ]
 [-122.7003585    45.58066533]]
sseSplit, and notSplit:  416.380767870588 1491.490570445267
the bestCentToSplit is:  2
the len of bestClustAss is:  27
[[-122.53564915   45.45881645]
 [-122.50376089   45.46110262]]
[[-122.57298327   45.52683995]
 [-122.48186875   45.49677212]]
[[-122.57463981   45.52861152]
 [-122.48812733   45.49597933]]
[[-122.57768158   45.53263337]
 [-122.49860291   45.49496564]]
sseSplit, and notSplit:  464.7205983452951 1056.432449734144
```

4. 对导入的实验数据，根据 sklearn 的 K-means 算法模块 sklearn.cluster.KMeans，完成对于 places.txt 数据的聚类

```
# bisecting K-means 二分K均值算法 克服局部最优值
def biKmeans(dataSet, k, distMeas=distEclud):
    m = shape(dataSet)[0]              # 样本个数
    clusterAssment = mat(zeros((m,2)))# 样本标记 分配结果 第一列索引 第二列误差
    centroid0 = mean(dataSet, axis=0).tolist()[0]# 创建一个初始质心
    centList =[centroid0] # 一个中心的 列表
    for j in range(m):    # 计算初始误差
        clusterAssment[j,1] = distMeas(mat(centroid0), dataSet[j,:])**2#每个样本与中心的距离平方
    while (len(centList) < k):# 中心数俩个未达到指定中心数量 继续迭代
        lowestSSE = inf       # 最小的 误差平方和 SSE
        for i in range(len(centList)):# 对于每一个中心
            ptsInCurrCluster = dataSet[nonzero(clusterAssment[:,0].A==i)[0],:] # 处于当前中心的样本点
            centroidMat, splitClustAss = kMeans(ptsInCurrCluster, 2, distMeas) # 对此中心内的点进行二分类
            # 该样本中心 二分类之后的 误差平方和 SSE
            sseSplit = sum(splitClustAss[:,1])
            # 其他未划分数据集的误差平方和 SSE
            sseNotSplit = sum(clusterAssment[nonzero(clusterAssment[:,0].A!=i)[0],1])
            print( "sseSplit, and notSplit: ",sseSplit,sseNotSplit)
            # 划分后的误差和没有进行划分的数据集的误差为本次误差
            if (sseSplit + sseNotSplit) < lowestSSE: # 小于上次 的 误差
                bestCentToSplit = i # 记录应该被划分的中心 的索引
                bestNewCents = centroidMat # 最好的新划分出来的中心
                bestClustAss = splitClustAss.copy()# 新中心 对于的 划分记录 索引(0或1)以及 误差平方
                lowestSSE = sseSplit + sseNotSplit # 更新总的 误差平方和
        # 记录中心 划分 数据
        bestClustAss[nonzero(bestClustAss[:,0].A == 1)[0],0] = len(centList)  # 现有中心数量
        bestClustAss[nonzero(bestClustAss[:,0].A == 0)[0],0] = bestCentToSplit# 最应该被划分的中心
        print ('the bestCentToSplit is: ',bestCentToSplit)
        print ('the len of bestClustAss is: ', len(bestClustAss))
        # 将最应该被划分的中心 替换为 划分后的 两个 中心(一个替换，另一个 append在最后添加)
        centList[bestCentToSplit] = bestNewCents[0,:].tolist()[0]# 替换
        centList.append(bestNewCents[1,:].tolist()[0])           # 添加
        # 更新 样本标记 分配结果 替换 被划分中心的记录
        clusterAssment[nonzero(clusterAssment[:,0].A == bestCentToSplit)[0],:]= bestClustAss
    return mat(centList), clusterAssment
```

```
####位置数据聚类测试#####
# 利用雅虎的服务器将地址转换为 经度和纬度
import urllib
import json
def geoGrab(stAddress, city):
    apiStem = 'http://where.yahooapis.com/geocode?'  #
    params = {}
    params['flags'] = 'J'          # 设置返回类型为JSON字符串
    params['appid'] = 'aaa0VN6k' # 注册 帐号后获得 http://developer.yahoo.com
    params['location'] = '%s %s' % (stAddress, city) # 位置信息
    url_params = urllib.urlencode(params)# 将字典转换成可以通过ＵＲＬ进行传递的字符串格式
    yahooApi = apiStem + url_params      # 加入网络地址
    print (yahooApi)                       # 打印 ＵＲＬ
    c=urllib.urlopen(yahooApi)           # 打开 ＵＲＬ
    return json.loads(c.read())          # 读取返回的jason字符串    对位置进行了编码 得到经度和纬度
```

```
from time import sleep
def massPlaceFind(fileName):
    fw = open('places.txt', 'w') # 打开位置信息文件
    for line in open(fileName).readlines():# 每一行
        line = line.strip()
        lineArr = line.split('\t')# 得到列表
        retDict = geoGrab(lineArr[1], lineArr[2])# 第二列为号牌 第三列为城市 进行地址解码
        if retDict['ResultSet']['Error'] == 0:
            lat = float(retDict['ResultSet']['Results'][0]['latitude']) #经度
            lng = float(retDict['ResultSet']['Results'][0]['longitude'])#纬度
            print ("%s\t%f\t%f" % (lineArr[0], lat, lng))
            fw.write('%s\t%f\t%f\n' % (line, lat, lng)) #再写入到文件
        else: print ("error fetching")
        sleep(1)#延迟1s
    fw.close()
```

```
# 返回地球表面两点之间的距离  单位英里 输入经纬度(度)  球面余弦定理
def distSLC(vecA, vecB):#Spherical Law of Cosines
    a = sin(vecA[0,1]*pi/180) * sin(vecB[0,1]*pi/180)
    b = cos(vecA[0,1]*pi/180) * cos(vecB[0,1]*pi/180) * \
                        cos(pi * (vecB[0,0]-vecA[0,0]) /180)
    return arccos(a + b)*6371.0 #pi  in numpy
```

```
import matplotlib
import matplotlib.pyplot as plt
import pylab
def clusterClubs(numClust=5):
    datList = []
    for line in open('places.txt').readlines():
        lineArr = line.split('\t')
        datList.append([float(lineArr[4]), float(lineArr[3])])
    datMat = mat(datList)
    myCentroids, clustAssing = biKmeans(datMat, numClust, distMeas=distSLC)
    fig = plt.figure()
    rect=[0.1,0.1,0.8,0.8]
    scatterMarkers=['s', 'o', '^', '8', 'p', \
                    'd', 'v', 'h', '>', '<']
    axprops = dict(xticks=[], yticks=[])
    ax0=fig.add_axes(rect, label='ax0', **axprops)
    imgP = plt.imread('Portland.png')
    ax0.imshow(imgP)
    ax1=fig.add_axes(rect, label='ax1', frameon=False) # 在同一张图上又创建一个字图
    for i in range(numClust): # 遍历每一个簇
        ptsInCurrCluster = datMat[nonzero(clustAssing[:,0].A==i)[0],:]
        markerStyle = scatterMarkers[i % len(scatterMarkers)]
        ax1.scatter(ptsInCurrCluster[:,0].flatten().A[0], ptsInCurrCluster[:,1].flatten().A[0], marker=markerStyle, s=90)
    ax1.scatter(myCentroids[:,0].flatten().A[0], myCentroids[:,1].flatten().A[0], marker='+', s=300)
    plt.show()


# 主函数
clusterClubs(5)
```

```
the len of bestClustAss is:  30
```