中山大学计算机学院人工智能本科生实验报告（2022学年春季学期）

课程名称：Artificial Intelligence

教学班级	专业（方向）	学号	姓名
2班	计算机科学与技术	21307174	刘俊杰

一、实验题目

Week 12 Kmeans算法实现

二、实验内容

实验内容

实验任务

- 在给定数据集完成k-means算法聚类，完成以下内容数据读取:文本形式读入行，split后转浮点数: 或使用numpy pandas 等库直接读取
- 设计合适的k以及距离度量函数并实现聚类算法k自选，度量函数自选 (如: 欧氏距离)
- 画出聚类后的数据可视化图,可使用matplotlib

1. 算法原理

k均值算法

用于硬聚类

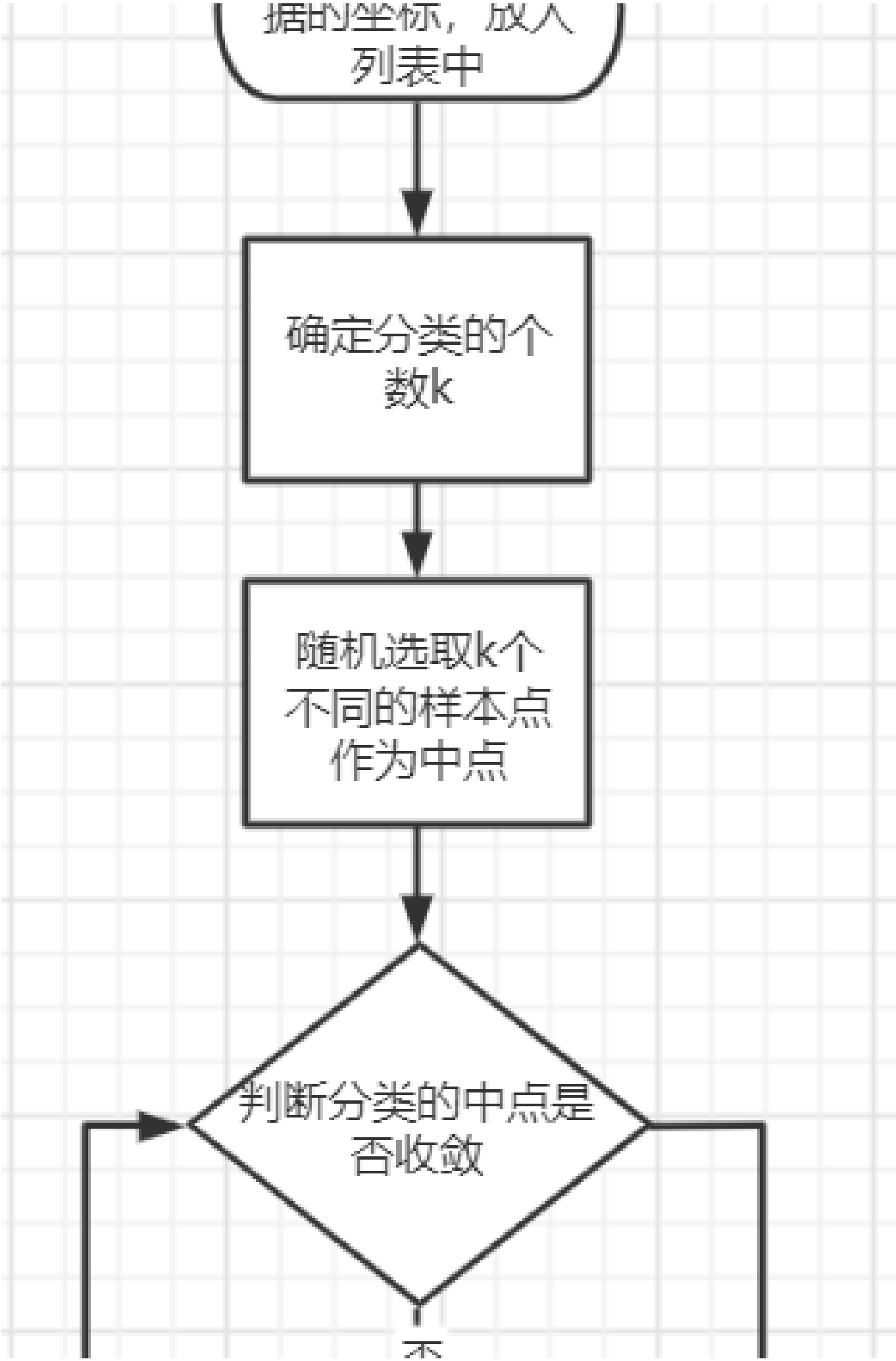
(1)选择k个重心 (centroid)

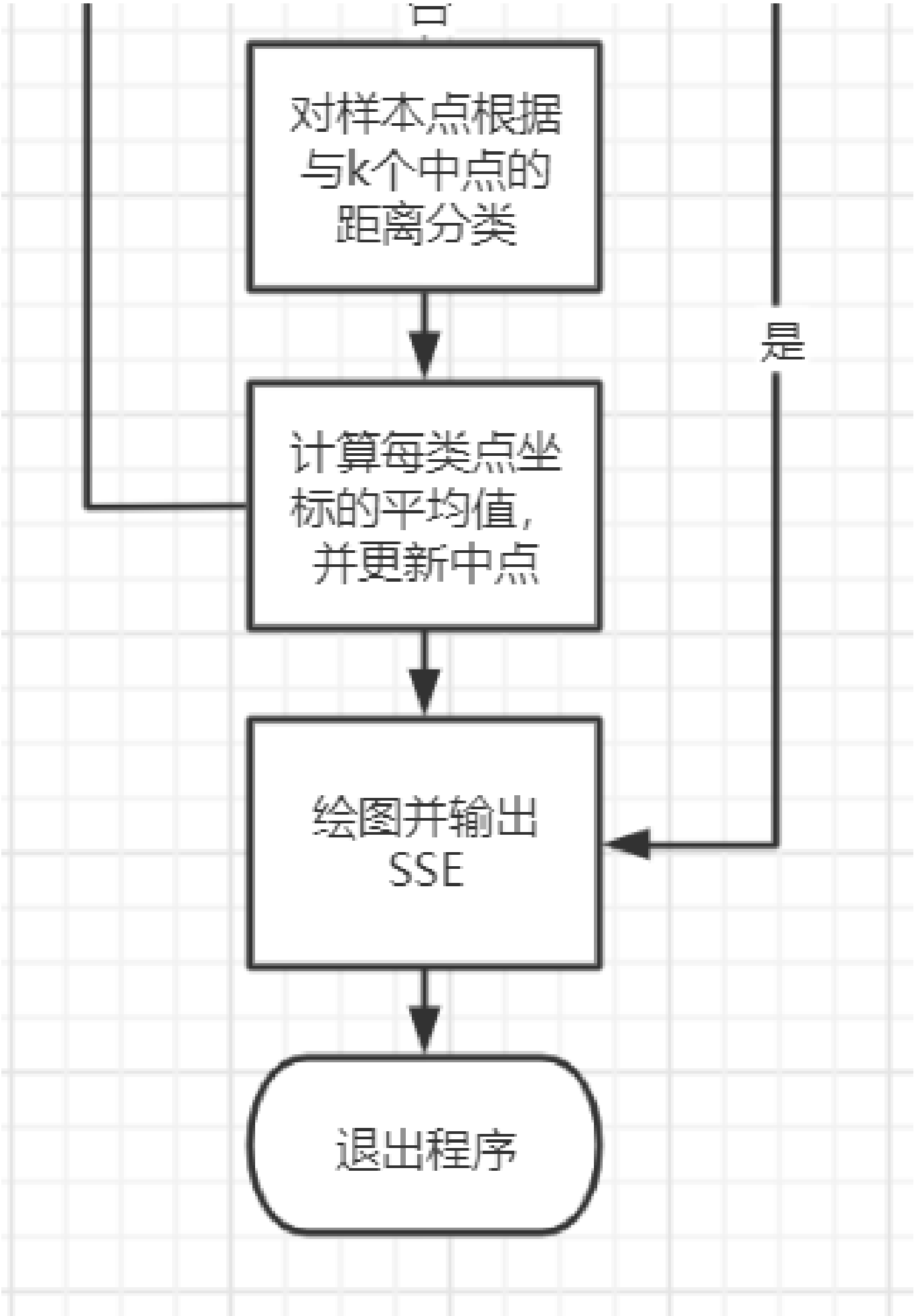
(2)寻找最近的重心并且更新聚类分配。将每个数据点都分配给离它最近的重心的聚类。距离的度量通常是欧式距离。

(3)将重心移动到它们的聚类的中心。每个聚类的重心的新位置是通过计算该聚类中所有数据点的平均位置得到的。

(4)重复第2和3步，直到每次迭代时重心的位置不再显著变化(即直到该算法收敛) 。

2.流程图


据的坐标，放入
列表中
确定分类的个
数k
随机选取k个
不同的样本点
作为中点
判断分类的中点是
否收敛



3.关键代码展示

(1)读取文件获得样本点坐标

```
def read_file(f):#读取文本信息
    data=[]#坐标
    x=[]#横坐标
    y=[]#纵坐标
    for line in f:
        if line=="X1,X2\n":
            continue
        line=line.strip().split(",")
        data.append([float(line[0]),float(line[1])])
        x.append(float(line[0]))
        y.append(float(line[1]))
    global N
    N=len(x)
    return data,x,y
```

(2)距离度量方式

```
def O_distance(A,B):#计算欧氏距离
    return math.sqrt(sum([(a - b)**2 for (a,b) in zip(A,B)]))
def O_distance_np(A,B):#np优化后的计算欧氏距离
    x1 = np.array(A)
    x2 = np.array(B)
    return np.sqrt(np.sum((x1 - x2)**2))
```

(3)最开始选择中点(一种随机选择，一种按PPT上通过距离算出的选择)

```
def choose_center_random(x,y,k):#初始随机选择样本点作为中点
    #k中点的数目
    has_choose=set()#set去重，防止随机选取
    has_chosen=[]#已选择作为中点的样本点(用下标表示)
    has_choose.add(-1)
    for i in range(k):
        choose_index=-1
        while(choose_index not in has_choose):##set去重
            choose_index=random.randint(0,N-1)#包含上下限：[a, b]
        has_choose.add(choose_index)
        has_chosen.append(choose_index)
    center=[]
    for i in range(k):
        center.append([x[i],y[i]])#记录中点
    return center

def choose_center_distance(x,y,k):#初始选择样本点作为中点
    #k中点的数目
    has_choose=set()#set去重，防止随机选取
    has_chosen=[]#已选择作为中点的样本点(用下标表示)
```

```
    has_choose.add(-1)
    distance=[0 for i in range(len(x))]
    while(len(has_chosen)<k):
        choose_index=-1
        for i in range(len(has_chosen)):
            distance_sum=0
            for j in range(len(x)):
                distance[j]=O_distance_np([x[has_chosen[i]],y[has_chosen[i]]],
[x[j],y[j]])
                distance[j]*=distance[j]
                distance_sum+=distance[j]
            for j in range(len(x)):
                distance[j]/=distance_sum#每个点选择作为中点的概率
            #选择中点
            while(choose_index in has_choose):##set去重
                pro=random.uniform(0,1)#包含上下限: [a, b]
                pro_tmp=0
                for j in range(len(x)):
                    if pro>=pro_tmp and pro<=pro_tmp+distance[j]:
                        pro_tmp+=distance[j]
                        choose_index=j
                        break
        has_choose.add(choose_index)
        has_chosen.append(choose_index)
    center=[]
    for i in has_chosen:
        center.append([x[i],y[i]])#记录中点
    return center
```

(4)通过计算样本点对k个点的距离，将样本点归类进距离最小的类

```
def classify(category,center,data):#分类
    category=[[] for i in range(len(center))]#category[i]记录第i类有哪些点
    for i in range(N):
        distance=[]#记录该点到每个中点的距离
        for j in range(len(center)):
            distance.append(O_distance_np(data[i],center[j]))#计算欧氏距离
        min_index=np.argsort(distance)[0]#将距离最近的中点的类作为我们对该点的归类
        category[min_index].append(i)
    return category
```

(5)计算每个类的各个点坐标的平均值作为新中点，更新中点

```
def calculate_center(category,center,data):#计算每个类的各个点坐标的平均值作为新中
点，更新中点
    n=0#统计需要更新的中点数目
    for i in range(len(center)):
        x_sum=0
        y_sum=0
```

```
        for j in range(len(category[i])):
            x_sum+=data[category[i][j]][0]
            y_sum+=data[category[i][j]][1]
        x_average=x_sum/(len(category[i]))#该类横坐标平均值
        y_average=y_sum/(len(category[i]))#该类纵坐标平均值
        if abs(x_average-center[i][0]) <mistake and abs(y_average-center[i][1])
<mistake:#若新旧中点误差小于规定的误差，则收敛
            n+=1
        else:#未收敛，则更新
            center[i]=[x_average,y_average]
    return n==len(center)#若全部收敛，则返回True
```

(6)绘图函数

```
def draw(category,center,data):#绘图
    final_x=[[] for i in range(len(center))]#每个类点的横坐标
    final_y=[[] for i in range(len(center))]#每个类点的纵坐标
    for i in range(len(center)):
        for j in range(len(category[i])):
            final_x[i].append(data[category[i][j]][0])
            final_y[i].append(data[category[i][j]][1])
    for i in range(len(center)):
        color=colours[i]#该类点的颜色
        plt.scatter([x for x in final_x[i]],[y for y in
final_y[i]],c=color,s=len([x for x in final_x[i]]))
        plt.scatter(center[i][0],center[i][1],marker='x',c="red")
    plt.show()
```

(7)计算SSE

```
def calculate_SSE(category,center,data):#计算分类后的SSE
    k_SSE=0
    for i in range(len(category)):
        for j in range(len(category[i])):
            k_SSE+=O_distance_np(center[i],data[category[i][j]])
    return k_SSE
```

(8)k_means算法

```
def k_means(data,x,y,k):#
    category=[[] for i in range(k)]#类
    center=choose_center_random(x,y,k)#初始随机选择样本点作为中点
    stop=False
    while(stop==False):#中点收敛则停止
        category=classify(category,center,data)#分类
        stop=calculate_center(category,center,data)#更新中点，中点收敛则stop=True
    k_SSE=calculate_SSE(category,center,data)#计算SSE
```

```
    print(k_SSE)#输出SSE
    draw(category,center,data)#绘图
```

4.创新点&优化

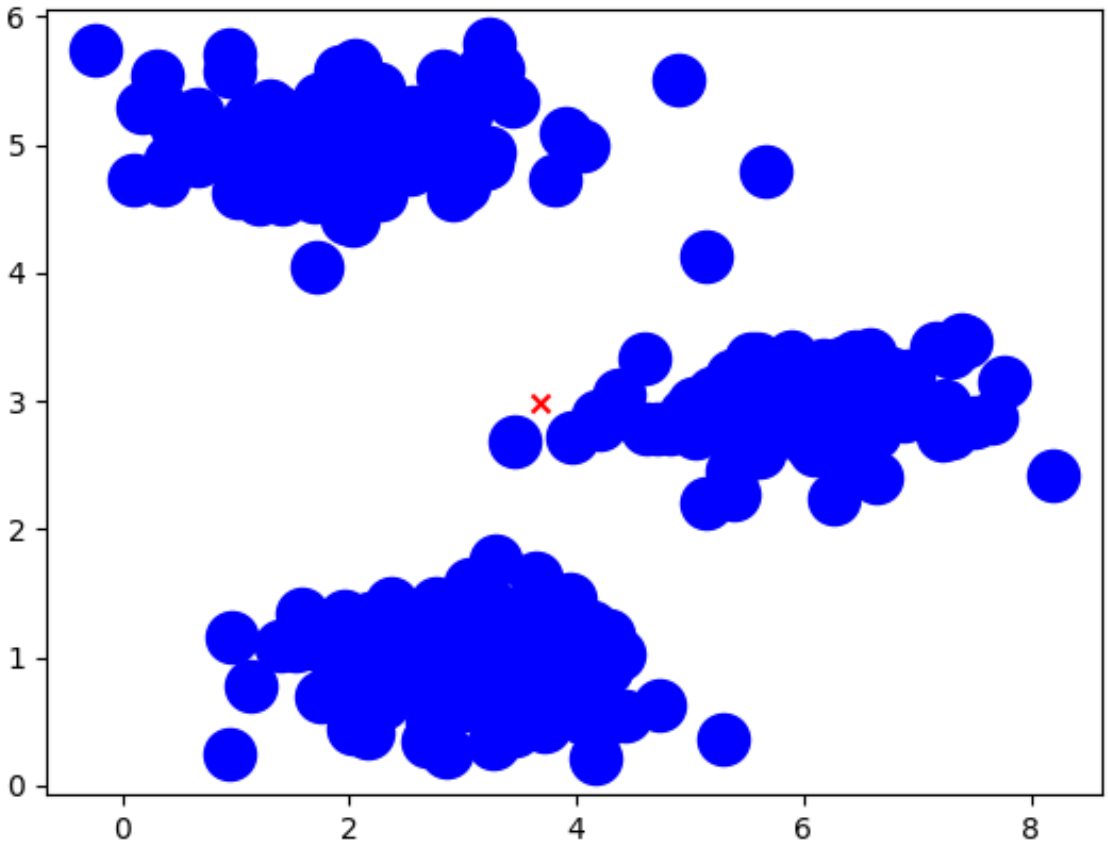
无

三、实验结果及分析

1. 实验结果展示示例(实验结果放入result文件夹中，这里展示的是初始中点选择方式为一开始随机选择中点的结果)

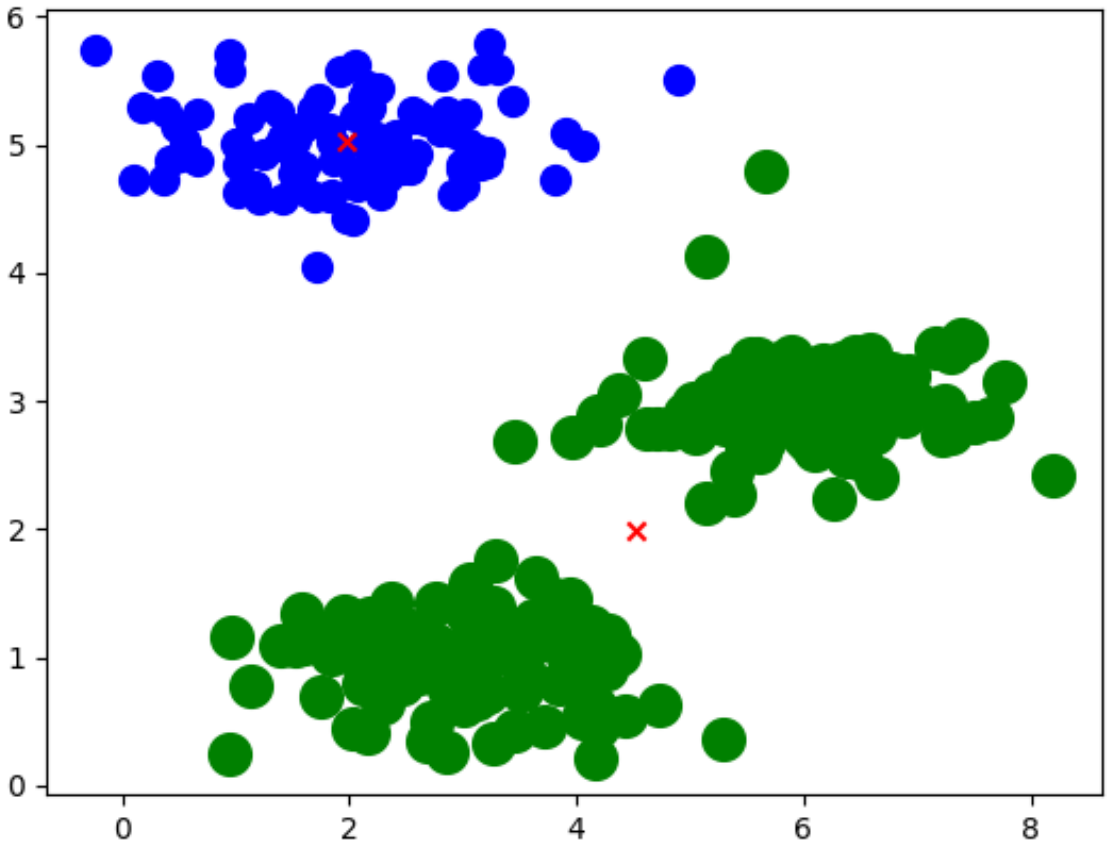
(1)k=1,SSE=737.9732545222748

分类图:



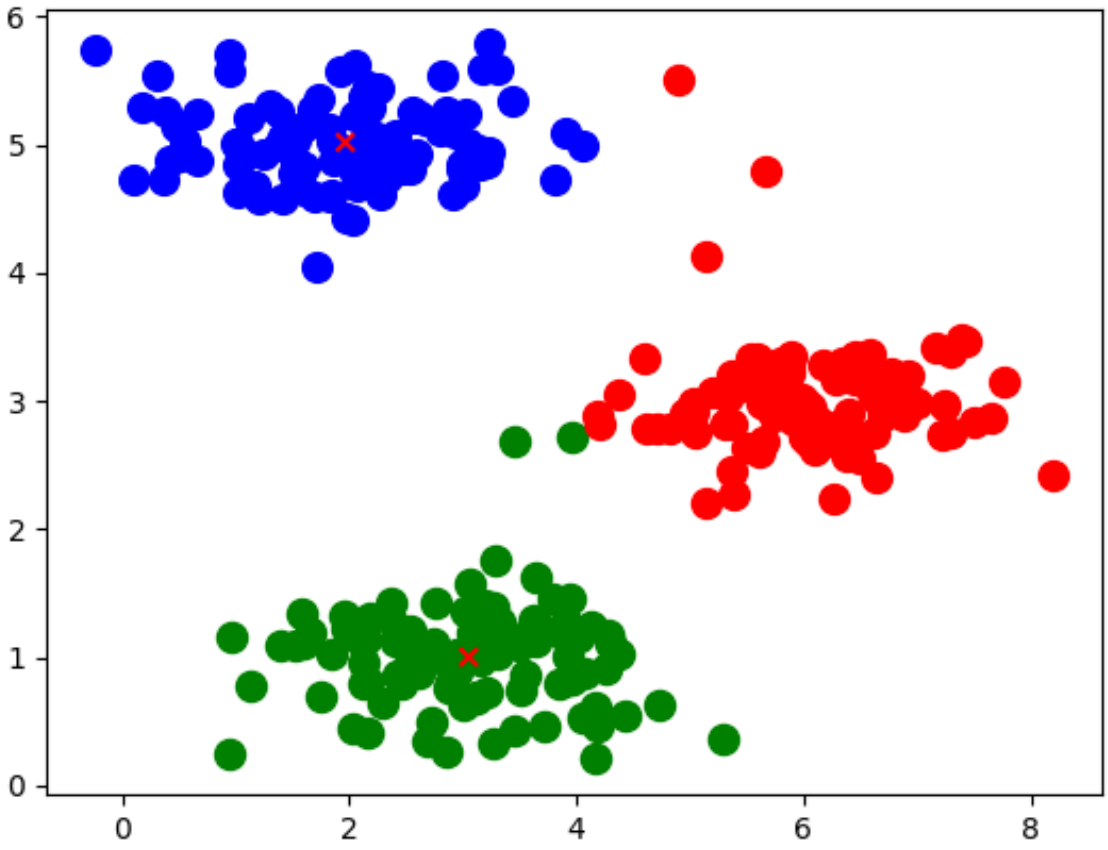
(2)k=2,SSE=464.1013125547029

分类图:



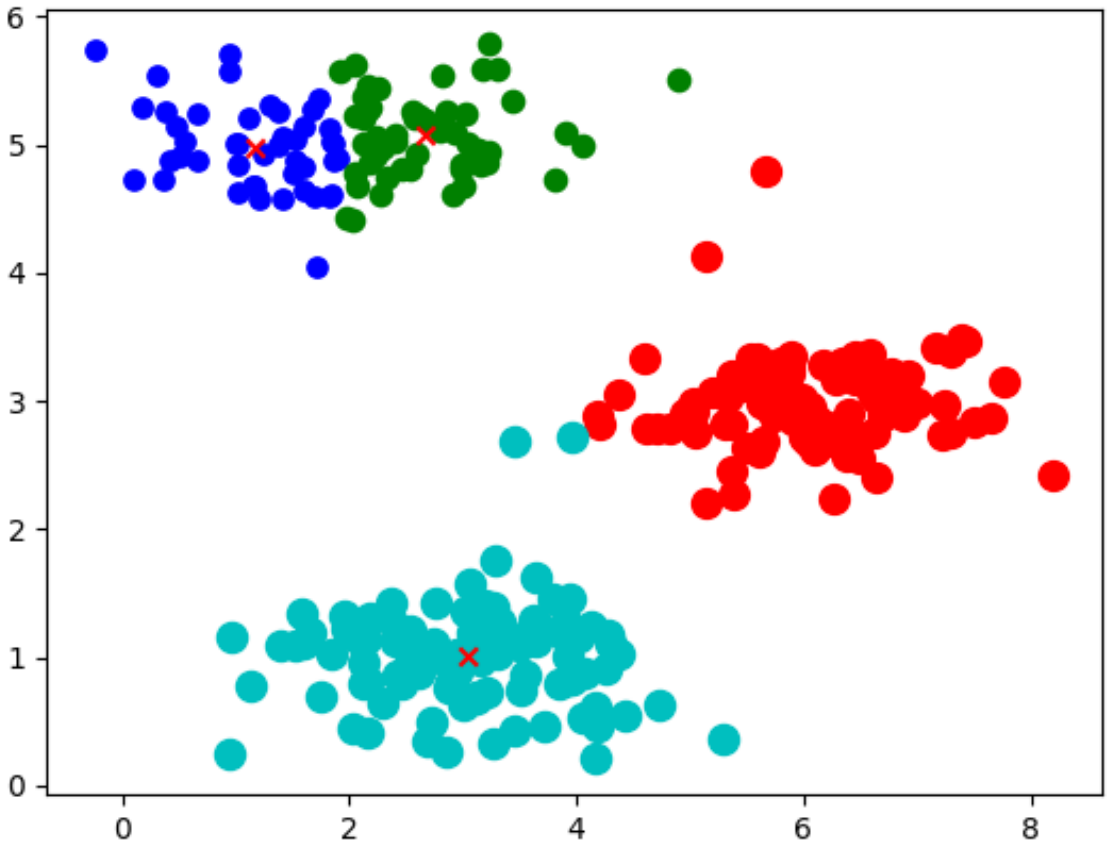
(3)k=3,SSE=238.25290901147605

分类图:



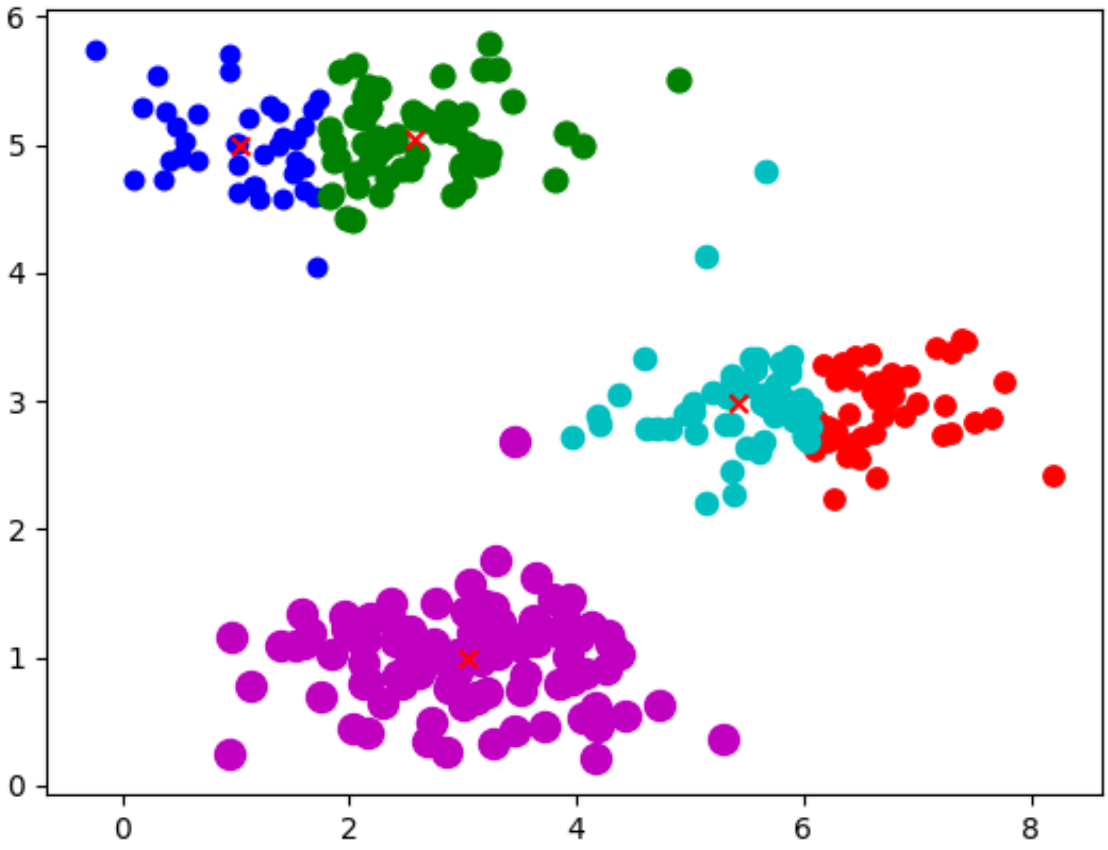
(4)k=4,SSE=212.02924330666679

分类图:



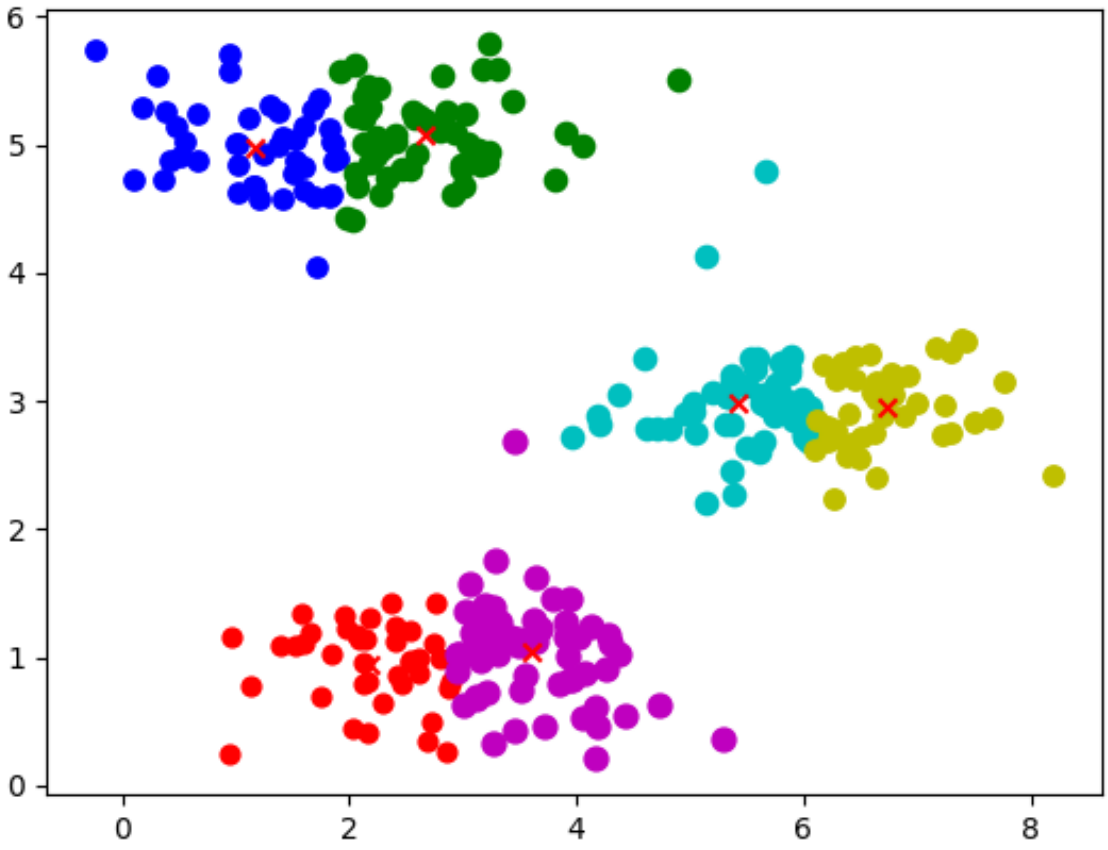
(5)k=5,SSE=205.73697276897983

分类图:



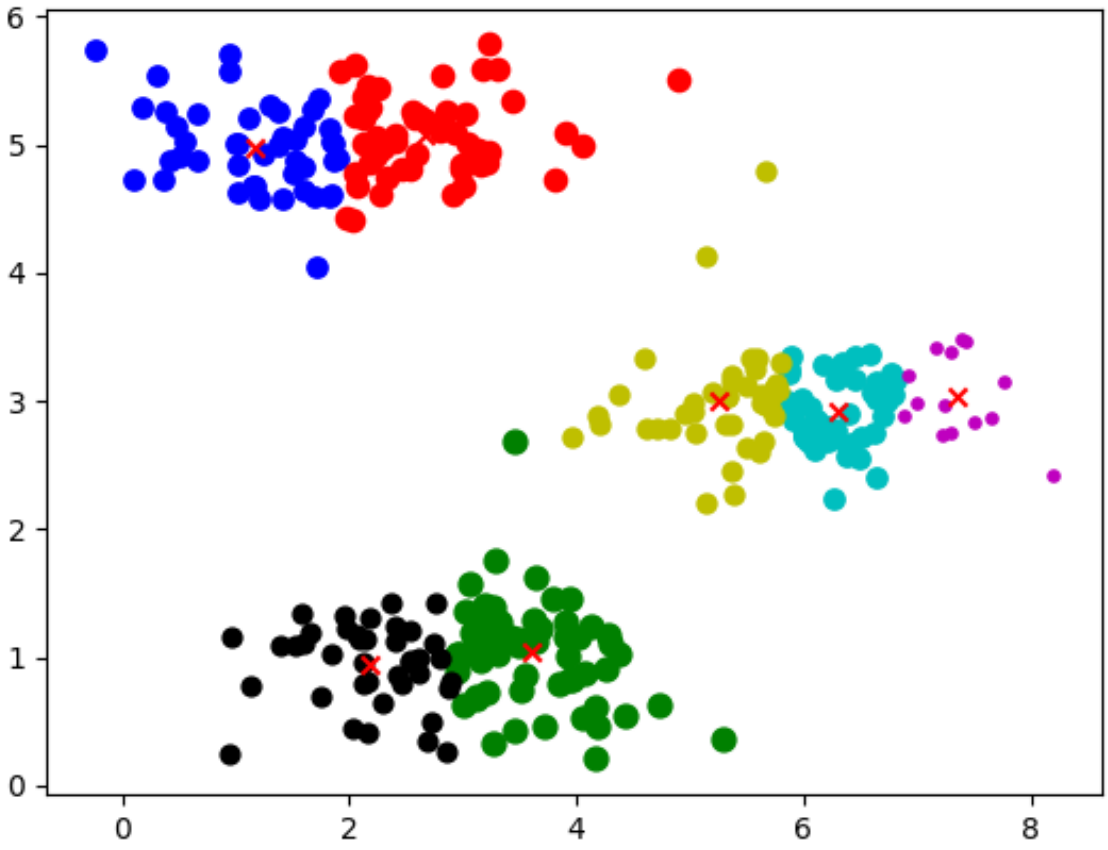
(6)k=6,SSE=195.38042154939035

分类图:



(7)k=7,SSE=152.08645524664877

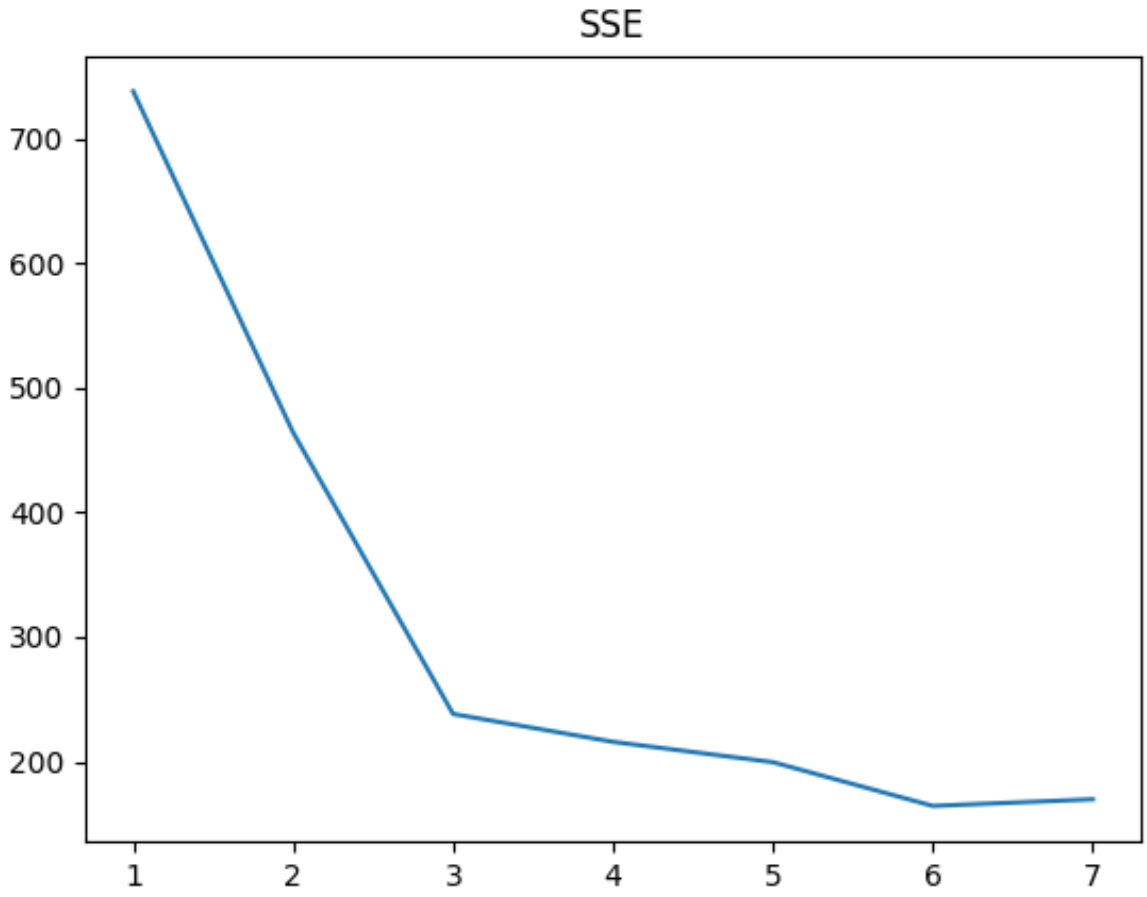
分类图:



2. 评测指标展示及分析

k(类的个数)	SSE(最开始随机选择中点)	SSE(一开始根据距离得出的概率选择中点)
1	737.9732545222748	737.9732545222748
2	464.1013125547029	464.1013125547029
3	238.25290901147605	238.25290901147622
4	212.02924330666679	212.02924330666679
5	205.73697276897983	189.88241783090322
6	195.38042154939035	182.21957630214123
7	152.08645524664877	156.88833870749735

SSE 图:



分析

通过对上述不同k值对应的SSE，我们可以看出随着聚类数k的增大，样本划分会更加精细，每个簇的聚合程度会逐渐提高，那么误差平方和SSE自然会逐渐变小。

当k小于3时，k的增大会大幅增加每个簇的聚合程度，故SSE的下降幅度会很大，而当k到3时，再增加k所得到的聚合程度回报会迅速变小，所以SSE的下降幅度会骤减，然后随着k值的继续增大而趋于平缓。

所以3就是数据的真实聚类数。

四、参考资料

实验课PPT