计算机与信息工程学院实验报告（**八**）

姓名：王赫 学号：1828070097 专业：数据科学与大数据技术(明德计划) 年级：2018级

课程：机器学习与数据挖掘 主讲教师：罗慧敏 辅导教师：\_\_\_\_\_\_\_

实验时间：2020年12月 23日 下午17时至18时，实验地点：606

实验题目： 编程实现k均值算法

实验目的： 掌握k均值聚类算法的推导过程和基本原理

实验环境（硬件和软件） Anaconda/Jupyter notebook/Pycharm

实验内容：

编码实现k均值算法，设置三组不同的k值、三组不同的初始中心点，在西瓜数据集4.0上进行实验比较，并讨论什么样的初始中心有利于取得好结果。

实验数据记录：

**要求：**

**一、**已经给定部分代码，补充完整的代码，需要补充代码的地方已经用红色字体标注，包括：

**（1）#补充随机初始化中心点的代码**

**（2）#补充计算数据点到中心点的距离，并判断该数据点所属中心点的代码；**

**（3）#补充k均值代码；**

**二、**将补充完整的代码提交，并提交实验结果；（**也可以自己重写这部分的代码提交**）

import numpy as np

import matplotlib.pyplot as plt

import matplotlib as mpl

import scipy.io

def loaddata():

data = np.loadtxt('watermelon\_4.txt',delimiter=',')

return data

X = loaddata()

plt.scatter(X[:, 0], X[:, 1], s=20)

#随机初始化中心点

def kMeansInitCentroids(X, k):

#从X的数据中随机取k个作为中心点

**# 补充随机初始化中心点的代码**

**idx = np.zeros(len(X)).reshape(X.shape[0],-1)**

**for i in range(len(X)):**

**minDistance = float('inf') # 初始为无限大**

**index = 0**

**for k in range(len(centroids)):**

**# 遍历求解欧几里得距离**

**dis = np.sum(np.power(X[i]-centroids[k], 2))**

**if dis < minDistance:**

**minDistance = dis**

**index = k**

**idx[i] = index**

**return idx**

#计算数据点到中心点的距离，并判断该数据点属于哪个中心点

def findClosestCentroids(X, centroids):

#idx中数据表明对应X的数据是属于哪一个中心点的

idx = np.zeros(len(X)).reshape( X.shape[0],-1)

for i in range(len(X)):

**#补充计算数据点到中心点的距离，并判断该数据点所属中心点的代码**

**#选择数据X中类别为k[i]的数据**

**data = X[np.where(idx==k[i])[0]]**

**#重新计算聚类中心**

**centroids[i] = np.sum(data,axis=0)/len(data)**

**return centroids**

#重新计算中心点位置

def computeCentroids(X, idx):

k = set(np.ravel(idx).tolist()) #找到所有聚类中心索引

k = list(k)

centroids = np.ndarray((len(k),X.shape[1]))

for i in range(len(k)):

#选择数据X中类别为k[i]的数据

data = X[np.where(idx==k[i])[0]]

#重新计算聚类中心

centroids[i] = np.sum(data,axis=0)/len(data)

return centroids

def k\_means(X, k, max\_iters):

initial\_centroids = kMeansInitCentroids(X,k)

**#补充k均值代码**

**for i in range(max\_iters):**

**if i == 0:**

**centroids = initial\_centroids**

**#计算样本到簇群中心的距离，并返回每个样本所属的簇中心**

**idx = findClosestCentroids(X, centroids)**

**#更新数据中心点**

**centroids = computeCentroids(X, idx)**

return idx,centroids

idx,centroids = k\_means(X, 3, 8)

print(idx)

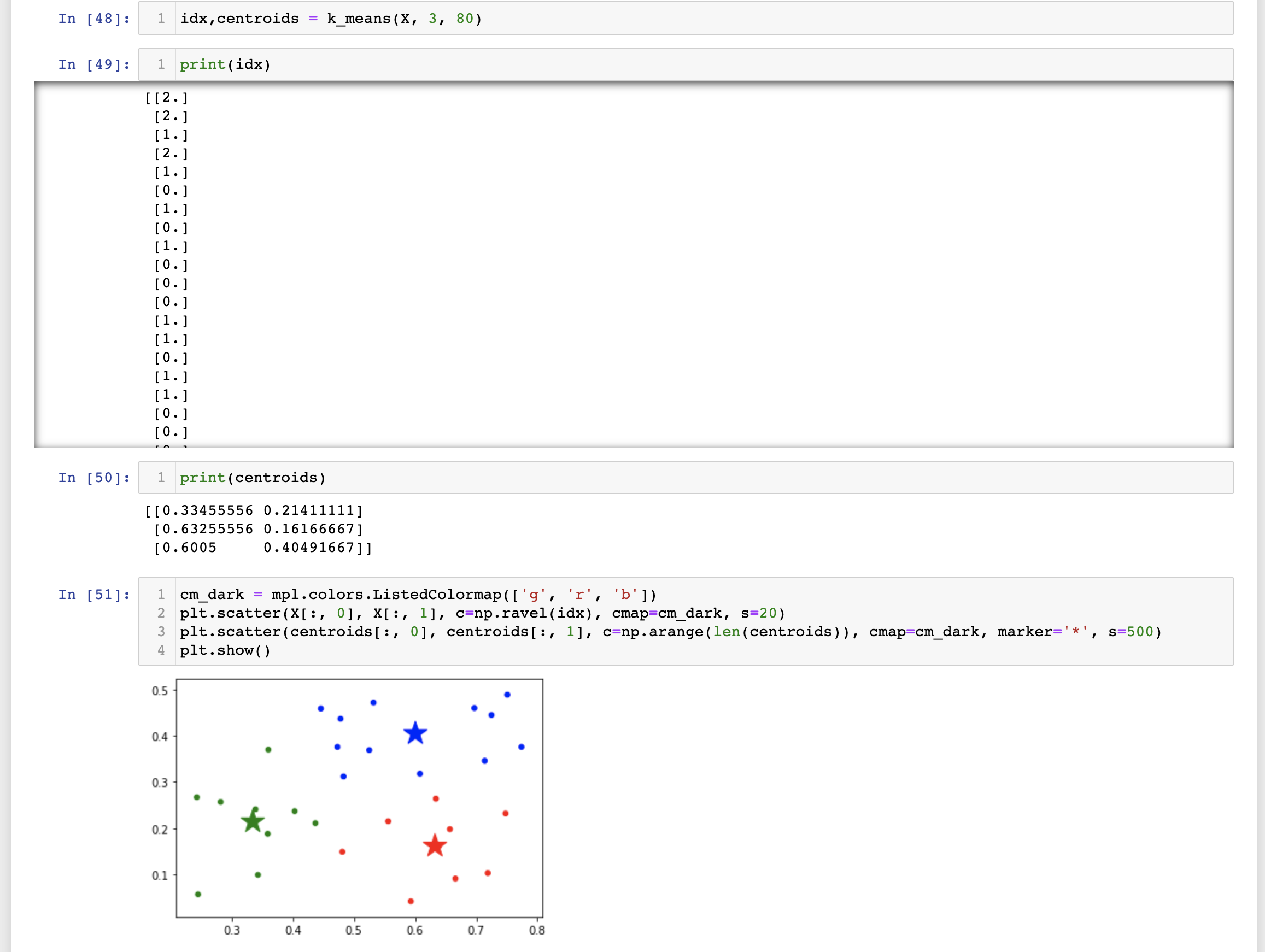
print(centroids)

cm\_dark = mpl.colors.ListedColormap(['g', 'r', 'b'])

plt.scatter(X[:, 0], X[:, 1], c=np.ravel(idx), cmap=cm\_dark, s=20)

plt.scatter(centroids[:, 0], centroids[:, 1], c=np.arange(len(centroids)), cmap=cm\_dark, marker='\*', s=500)

plt.show()



问题讨论：

本次实验主要进行的是K-Means算法编程实现，算法实现的核心主要是聚合中心点的迭代选择，然后计算各点到各簇中心点的距离，实验中主要采用的是欧氏距离。

和前面的学习一样，在进行多分类学习时，需要注意迭代次数的增加。