P69页：习题1第2个问题源代码：

import matplotlib.pyplot as plt

import numpy as np

from sklearn.cluster import KMeans

from sklearn.datasets import make\_moons

#调用sklearn中的make\_moons形成数据集

plt.subplot(122)

x1,y1=make\_moons(n\_samples=1000,noise=0.1)

#绘制数据图像

plt.title('make\_moons function example')

plt.scatter(x1[:,0],x1[:,1],marker='o')

~~#plt.scatter(x1[:,0],x1[:,1],marker='o',c=y1)~~

plt.show()

#用K\_均值聚类方法来做聚类，首先选择k=2

clusters=2

kmeans = KMeans(clusters,init="random")#构造聚类器

y\_pred=kmeans.fit\_predict(x1) #聚类

#输出质心以及每个原始数据对应类别

print("the centroid of cluster are:\n{0}".format(kmeans.cluster\_centers\_))#输出质心

print("the member of cluster are:{0}\n".format(kmeans.labels\_))#输出每个原始数据对应类别

#绘制聚类后的散点图

plt.scatter(x1[:,0],x1[:,1],c=y\_pred)

plt.show()

P72页 习题2第二个任务源代码：

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  # X\_train5 为样本点  X\_train5 = np.array([[2, 1],[3, 2],[4, 2],[1, 3],[1.5, 4],[1.7, 3],[2.6, 5],[3.4, 3],  [3, 6],[1, 7],[4, 5],[1.2, 6],[1.8, 7],[2.2, 8],[3.7, 7],[4.8, 5],[3.6, 3],[2.4, 2],[1.2, 3],[4.9, 1.5]])  # y\_train 为样本点标记  y\_train = np.array([1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5])  plt.scatter(X\_train5[:,0], X\_train5[:,1], c='g', marker='o', label='train\_class')  plt.show()  # X\_test 为测试样本  X\_test = np.array([3.2, 5.4])  # k 为邻居数  k = 5  # 这里的距离公式采用欧式距离  square\_ = (X\_train - X\_test) \*\* 2  square\_sum = square\_.sum(axis=1) \*\* 0.5  square\_sum\_sort = square\_sum.argsort()  small\_k = square\_sum\_sort[:k]  y\_test\_sum = np.bincount(np.array([y\_train[i] for i in small\_k])).argsort()  print('predict: class {}'.format(y\_test\_sum[-1]))  # 将数据可视化 更生动形象，数据分五类  # 将 class1 一类的样本点 放到 X\_train\_0中  X\_train\_1 = np.array([X\_train5[i, :] for i in range(len(y\_train)) if y\_train[i] == 1])  # 将 class2 一类的样本点 放到 X\_train\_2中  X\_train\_2 = np.array([X\_train5[i, :] for i in range(len(y\_train)) if y\_train[i] == 2])  # 将 class3 一类的样本点 放到 X\_train\_3中  X\_train\_3 = np.array([X\_train5[i, :] for i in range(len(y\_train)) if y\_train[i] == 3])  # 将 class4 一类的样本点 放到 X\_train\_4中  X\_train\_4 = np.array([X\_train5[i, :] for i in range(len(y\_train)) if y\_train[i] == 4])  # 将 class5 一类的样本点 放到 X\_train\_5中  X\_train\_5 = np.array([X\_train5[i, :] for i in range(len(y\_train)) if y\_train[i] == 5])  # 绘制所有样本点 并采用不同的颜色 分别标记  plt.scatter(X\_train\_1[:,0], X\_train\_1[:,1], c='g', marker='o', label='train\_class1')  plt.scatter(X\_train\_2[:,0], X\_train\_2[:,1], c='m', marker='o', label='train\_class2')  plt.scatter(X\_train\_3[:,0], X\_train\_3[:,1], c='y', marker='o', label='train\_class3')  plt.scatter(X\_train\_4[:,0], X\_train\_4[:,1], c='c', marker='o', label='train\_class4')  plt.scatter(X\_train\_5[:,0], X\_train\_5[:,1], c='k', marker='o', label='train\_class5')  if y\_test\_sum[-1] == 1:  test\_class = 'g'  elif y\_test\_sum[-1] == 2:  test\_class = 'm'  elif y\_test\_sum[-1] == 3:  test\_class = 'y'  elif y\_test\_sum[-1] == 4:  test\_class = 'c'  elif y\_test\_sum[-1] == 5:  test\_class = 'k'    plt.scatter(X\_test[0], X\_test[1], c=test\_class, marker='\*', s=100, label='test\_class')  # 连接 测试样本 与k个近邻  for i in small\_k:  plt.plot([X\_test[0], X\_train[i, :][0]], [X\_test[1], X\_train[i, :][1]], c='c')  plt.legend(loc='best')  plt.show() |