1、

(1)读取dating\_TestSet2.txt数据，将前三列作为data部分，最后一列作为label 部分，

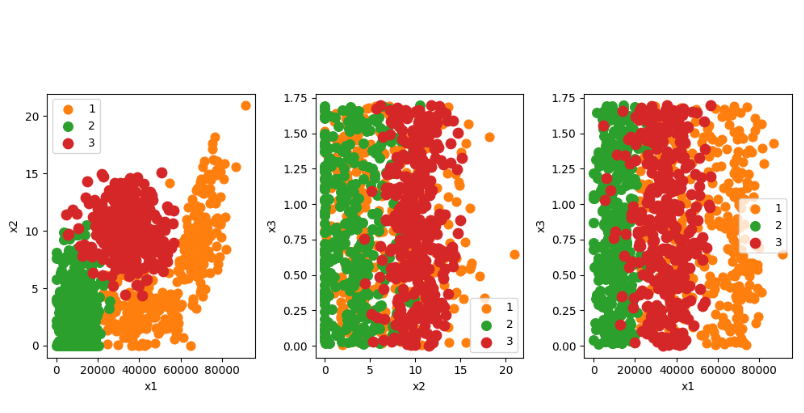
(2)使用KNN算法进行训练，选取其中一个样本进行预测，输出预测的结果(1/2/3)

import operator  
import numpy as np  
  
  
def knn(inX, dataSet, labels, k):  
 dataSetSize = dataSet.shape[0] # 行数  
 # print(dataSetSize)  
 # 计算欧式距离  
 diffMat = np.tile(inX, (dataSetSize, 1))-dataSet # 扩展dataSet行，分别相减，形成（x1-x2）矩阵  
 sqDiffMat = diffMat\*\*2  
 sqDistances = sqDiffMat.sum(axis=1)  
 distances = sqDistances\*\*0.5  
 # print(type(distances))  
 sortedDistIndicies = distances.argsort() # 从小到大排序，获得索引值(下标）  
 # print(sortedDistIndicies)  
 # 选择距离最小的k个点  
 classCount = {}#声明了一个字典  
 for i in range(k):  
 voteIlabel = labels[sortedDistIndicies[i]]  
 classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1  
 # for key,value in classCount.items():  
 # print(key,value)  
 # 排序  
 sortedClassCount = sorted(  
 classCount.items(), key=operator.itemgetter(1), reverse=True)  
 return int(sortedClassCount[0][0])  
  
  
# 读取数据  
data = np.loadtxt(  
 "D:\\Desktop\\python\\ArtificialIntelligence\\datingTestSet2.txt", delimiter=" ")  
  
# 分割数据  
label = data[:, -1]  
data = data[:, :3]  
# print(data)  
  
# 选择一个样本进行预测  
sample = data[0]  
  
# 预测结果  
prediction = knn(sample, data, label, k=10)  
  
# 输出预测结果  
print(f"预测结果：{prediction}")



2、数据可视化，根据标签，分别对1、2维 ；2，3维 ；1，3维 进行样本特征可视化plt.scatter(x1,x2,s=,color='')

import matplotlib.pyplot as plt  
import numpy as np  
  
data = np.loadtxt(  
 "D:\\Desktop\\python\\ArtificialIntelligence\\datingTestSet2.txt", delimiter=" ")  
  
# print(data.shape)  
# 数据集特征  
x1 = data[:, 0] # 第一个维度  
x2 = data[:, 1] # 第二个维度  
x3 = data[:, 2] # 第三个维度  
# print(x1)  
# 样本标签  
labels = data[:, 3]  
# print(labels)  
# 绘制 1、2 维的散点图  
plt.figure(figsize=(10, 4))  
plt.subplot(1, 3, 1)  
for label in set(labels):  
 plt.scatter([x1[i] for i in range(len(labels)) if labels[i] == label],  
 [x2[i] for i in range(len(labels)) if labels[i] == label],  
 label=int(label),  
 c='C{}'.format(int(label)), # 使用不同的颜色，C0, C1, C2, ...  
 s=50 + 10 \* label) # 设置不同标签下散点的大小不同  
plt.xlabel('x1')  
plt.ylabel('x2')  
plt.legend()  
# plt.show()  
  
# 绘制 2、3 维的散点图  
plt.subplot(1, 3, 2)  
for label in set(labels):  
 plt.scatter([x2[i] for i in range(len(labels)) if labels[i] == label],  
 [x3[i] for i in range(len(labels)) if labels[i] == label],  
 label=int(label),  
 c='C{}'.format(int(label)), # 使用不同的颜色，C0, C1, C2, ...  
 s=50 + 10 \* label) # 设置不同标签下散点的大小不同  
plt.xlabel('x2')  
plt.ylabel('x3')  
plt.legend()  
  
# 绘制 1、3 维的散点图  
plt.subplot(1, 3, 3)  
for label in set(labels):  
 plt.scatter([x1[i] for i in range(len(labels)) if labels[i] == label],  
 [x3[i] for i in range(len(labels)) if labels[i] == label],  
 label=int(label),  
 c='C{}'.format(int(label)), # 使用不同的颜色，C0, C1, C2, ...  
 s=50 + 10 \* label) # 设置不同标签下散点的大小不同  
plt.xlabel('x1')  
plt.ylabel('x3')  
plt.legend()  
  
plt.tight\_layout()  
plt.show()

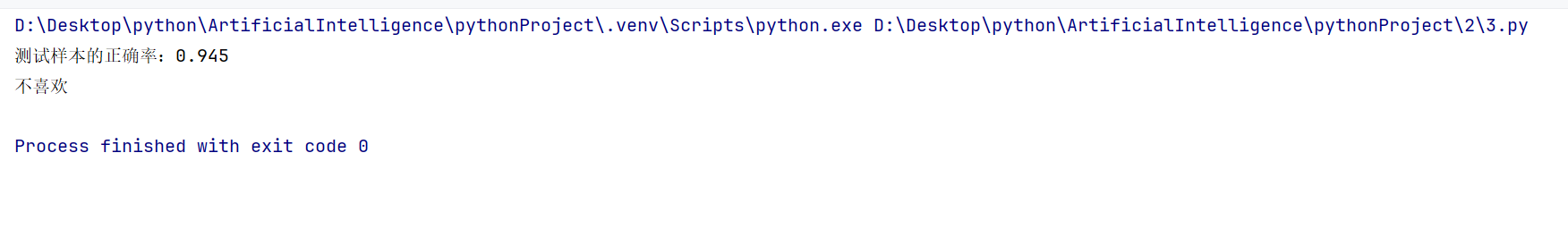


3、

(1)在第一题的基础上，对数据部分进行归一化处理，使用数据的80%进行训练，20%进行测试，输出测试样本的正确率，

(2)创建一个新的样本(也进行归一化处理)，输出预测值（['不喜欢','魅力一般','极具魅力']）

import operator  
  
import numpy as np  
  
  
def knn(inX, dataSet, labels, k):  
 dataSetSize = dataSet.shape[0] # 行数  
 # print(dataSetSize)  
 # 计算欧式距离  
 diffMat = np.tile(inX, (dataSetSize, 1)) - dataSet # 扩展dataSet行，分别相减，形成（x1-x2）矩阵  
 sqDiffMat = diffMat \*\* 2  
 sqDistances = sqDiffMat.sum(axis=1)  
 distances = sqDistances \*\* 0.5  
 # print(type(distances))  
 sortedDistIndicies = distances.argsort() # 从小到大排序，获得索引值(下标）  
 # print(sortedDistIndicies)  
 # 选择距离最小的k个点  
 classCount = {} # 声明了一个字典  
 for i in range(k):  
 voteIlabel = labels[sortedDistIndicies[i]]  
 classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1  
 # for key,value in classCount.items():  
 # print(key,value)  
 # 排序  
 sortedClassCount = sorted(  
 classCount.items(), key=operator.itemgetter(1), reverse=True)  
 return int(sortedClassCount[0][0])  
  
def autoNorm(dataSet):  
 minVals = dataSet[:, :-1].min(0) # 每列的最小值  
 maxVals = dataSet[:, :-1].max(0) # 每列的最大值  
 ranges = maxVals - minVals # 取值范围  
 normDataSet = (dataSet[:, :-1] - minVals) / ranges # 归一化的数据集  
 normDataSet = np.hstack((normDataSet, dataSet[:, -1][:, np.newaxis]))  
 return normDataSet,minVals,maxVals  
  
def splitDataSet(dataSet, ratio=0.8):  
 m = dataSet.shape[0] # 数据集样本数量  
 numTrain = int(m \* ratio) # 训练集样本数量  
 trainData = dataSet[:numTrain, :-1] # 训练集数据  
 testData = dataSet[numTrain:, :-1] # 测试集数据  
 trainLabels = dataSet[:numTrain, -1] # 训练集标签  
 testLabels = dataSet[numTrain:, -1] # 测试集标签  
 return trainData, trainLabels, testData, testLabels  
  
# 读取数据  
data = np.loadtxt("D:\\Desktop\\python\\ArtificialIntelligence\\datingTestSet2.txt", delimiter=" ")  
# print(data)  
# 归一化处理  
normDataSet,minVals,maxVals = autoNorm(data)  
# print(normDataSet)  
# 划分训练集和测试集  
trainData, trainLabels, testData, testLabels = splitDataSet(normDataSet)  
# print(trainData)  
# print(trainLabels)  
  
# 计算测试样本的正确率  
numTest = testData.shape[0]  
errorCount = 0  
for i in range(numTest):  
 predictLabel = knn(testData[i], trainData, trainLabels, k=10)  
 if predictLabel != testLabels[i]:  
 errorCount += 1  
accuracy = 1 - errorCount / float(numTest)  
print(f"测试样本的正确率：{accuracy}")  
  
# 预测新样本  
newSample=np.array([70000,8,1])  
normSample = (newSample - minVals)/(maxVals-minVals)  
# print(normSample)  
label\_map = {1: '不喜欢', 2: '魅力一般', 3: '极具魅力'}  
predictLabel = knn(normSample,trainData, trainLabels, k=10)  
print(label\_map[predictLabel])



4、读取iris.csv的数据，使用KNN算法对90%的数据进行训练，对10%的数据进行测试，输出测试集预测的错误率，选择一个样本输出预测的结果。

import operator  
  
import pandas as pd  
import numpy as np  
  
def knn(inX, dataSet, labels, k):  
 dataSetSize = dataSet.shape[0] # 行数  
 # print(dataSetSize)  
 # 计算欧式距离  
 diffMat = np.tile(inX, (dataSetSize, 1))-dataSet # 扩展dataSet行，分别相减，形成（x1-x2）矩阵  
 sqDiffMat = diffMat\*\*2  
 sqDistances = sqDiffMat.sum(axis=1)  
 distances = sqDistances\*\*0.5  
 # print(type(distances))  
 sortedDistIndicies = distances.argsort() # 从小到大排序，获得索引值(下标）  
 # print(sortedDistIndicies)  
 # 选择距离最小的k个点  
 classCount = {}#声明了一个字典  
 for i in range(k):  
 voteIlabel = labels[sortedDistIndicies[i]]  
 classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1  
 # for key,value in classCount.items():  
 # print(key,value)  
 # 排序  
 sortedClassCount = sorted(  
 classCount.items(), key=operator.itemgetter(1), reverse=True)  
 return sortedClassCount[0][0]  
  
def splitDataSet(dataSet, ratio=0.8):  
 # 将DataFrame转换为Numpy数组并保留索引和标签  
 # array = df.to\_numpy(index=False, columns=False)  
 dataSet=dataSet.sample(frac=1, random\_state=42) # 随机打乱数据集  
 # print(dataSet)  
 dataSet=dataSet.values  
 # print(dataSet)  
 m = dataSet.shape[0] # 数据集样本数量  
 numTrain = int(m \* ratio) # 训练集样本数量  
 trainData = dataSet[:numTrain, 1:-1] # 训练集数据  
 testData = dataSet[numTrain:, 1:-1] # 测试集数据  
 # print(trainData)  
 # print(testData)  
 trainLabels = dataSet[:numTrain, -1] # 训练集标签  
 testLabels = dataSet[numTrain:, -1] # 测试集标签  
 return trainData, trainLabels, testData, testLabels  
  
# 读取数据  
data = pd.read\_csv(r"D:\Desktop\python\ArtificialIntelligence\pythonProject\iris.csv",delimiter=',')  
  
# 划分训练集和测试集  
trainData, trainLabels, testData, testLabels = splitDataSet(data,0.9)  
  
# 计算测试集的预测错误率  
# 计算测试样本的正确率  
numTest = testData.shape[0]  
errorCount = 0  
for i in range(numTest):  
 predictLabel = knn(testData[i], trainData, trainLabels, k=10)  
 if predictLabel != testLabels[i]:  
 errorCount += 1  
accuracy = 1 - errorCount / float(numTest)  
error\_rate = 1 - accuracy  
print("测试集预测错误率：", error\_rate)  
  
# 选择一个样本输出预测结果  
sample\_index = 10  
# print(testData.shape)  
sample = testData[sample\_index]  
predicted\_label = knn(sample,trainData, trainLabels, k=10)  
print("样本的预测结果：", predicted\_label)  
print("正确结果",testLabels[sample\_index])

