机器学习

上机实验报告

学部： 计算机与信息工程学院

专业：电子信息

班级：电信研A1

学号：20241513007

姓名：梅少伟

日期：2024.4.18

实验五 Knn算法实验

【实验目的】

编程验证教材P311例8.12，用python实现KNN算法，并对约会数据集datingTestSet.txt的数据进行分析，训练分类器。分类器训练好之后，输入“每年获得的飞行常客里程数”，“玩视频游戏所消耗时间百分比”，“每周消费的冰激淋升数”这3个特征值来预测你对这个人的印象：“讨厌”，“有些喜欢”或“非常喜欢”。注意，datingTestSet.txt数据集和教材代码里的数据不一样，特征排序不一样，datingTestSet.txt数据集里的特征顺序：每年获得的飞行常客里程数，玩视频游戏所消耗时间百分比，每周消费的冰激淋，对这个人的印象；教材里的代码默认顺序：玩视频游戏所消耗时间百分比，每年获得的飞行常客里程数，每周消费的冰激淋升数，对这个人的印象，即把“每年获得的飞行常客里程数”和“玩视频游戏所消耗时间百分比”两者的顺序交换了。

【实验性质】

验证型实验

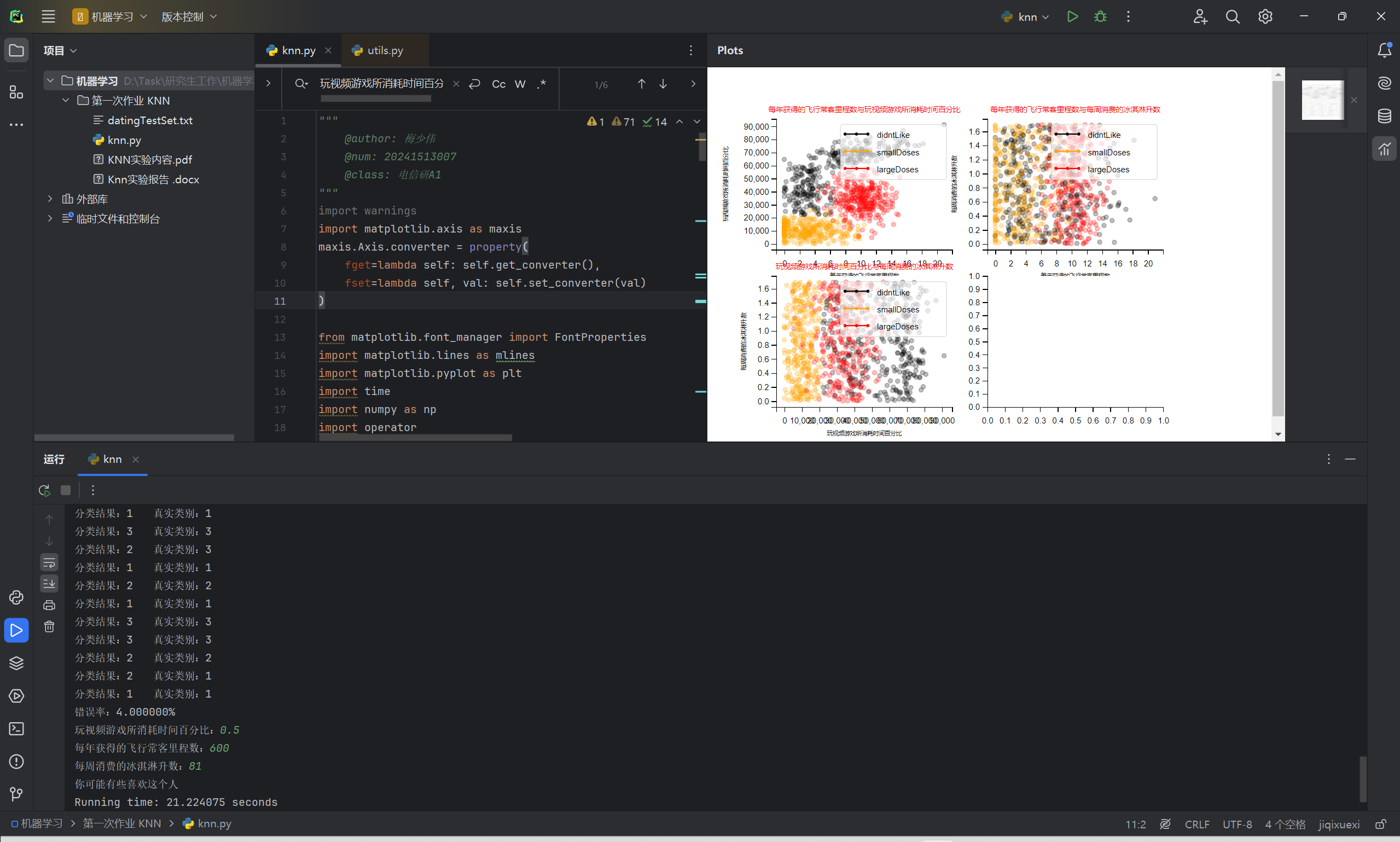
【实验内容】

    1、knn算法。

【实验环境】

Python

1. 实验结果的截图粘贴在答案框，



1. 实验报告书作为附件上传

"""

@author: 梅少伟

@num: 20241513007

@class: 电信研A1

"""

import warnings

import matplotlib.axis as maxis

maxis.Axis.converter = property(

fget=lambda self: self.get\_converter(),

fset=lambda self, val: self.set\_converter(val)

)

from matplotlib.font\_manager import FontProperties

import matplotlib.lines as mlines

import matplotlib.pyplot as plt

import time

import numpy as np

import operator

def classify0(inX, dataSet, labels, k):

# KNN算法，分类器

dataSetSize = dataSet.shape[0] # shape[0]返回dataSet的行数

diffMat = np.tile(inX, (dataSetSize, 1)) - dataSet

# 将inx重复dataSetSize次并排成一列

sqDiffMat = diffMat \*\* 2

# 二维特征相减后平方

sqDistances = sqDiffMat.sum(axis=1)

# sum将所有元素相加，sum(0)将所有列相加，sum(1)将所有行相加

distances = sqDistances \*\* 0.5

# 开方，计算距离

sortedDistIndices = distances.argsort()

# argsort函数返回的是距离值从小到大的索引

classCount = {}

# 定义一个记录同一类别中的样本个数的字典

for i in range(k):

# 选择距离最小的k个点

voteIlabel = labels[sortedDistIndices[i]]

classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1

sortedClassCount = sorted(classCount.items(),

key=operator.itemgetter(1), reverse=True)

return sortedClassCount[0][0]

# 返回样本个数最多的类别，即待分类样本的类别

def file2matrix(filename):

# 打开解析文件，对数据进行分类

fr = open(filename)

arrayOlines = fr.readlines()

numberOfLines = len(arrayOlines)

returnMat = np.zeros((numberOfLines, 3))

# 返回矩阵中的numberOfLines行、3列

classLabelVector = []

# 创建分类标签向量

index = 0

for line in arrayOlines:

# 读取每一行

line = line.strip()

# 去掉每一行首尾的空白符，例如"\n"，"\t"，"!!"

listFromLine = line.split("\t")

# 由于文件中“飞行常客里程数”和“视频游戏百分比”已对调，这里交换读取顺序：

# 第一列：玩视频游戏所消耗时间百分比 (原来是第2列)

returnMat[index, 0] = float(listFromLine[1])

# 第二列：每年获得的飞行常客里程数 (原来是第1列)

returnMat[index, 1] = float(listFromLine[0])

# 第三列：每周消费的冰淇淋升数

returnMat[index, 2] = float(listFromLine[2])

# 根据文本内容进行分类：1为讨厌，2为喜欢，3为非常喜欢

if listFromLine[-1] == 'didntLike':

classLabelVector.append(1)

elif listFromLine[-1] == 'smallDoses':

classLabelVector.append(2)

elif listFromLine[-1] == 'largeDoses':

classLabelVector.append(3)

index += 1

return returnMat, classLabelVector

# 返回特征矩阵以及分类标签向量

def showdatas(datingDataMat, datingLabels):

# 可视化数据

# 设置汉字为14号简体字

font = FontProperties(fname=r"C:\windows\Fonts\simsun.ttc", size=14)

fig, axs = plt.subplots(nrows=2, ncols=2, sharex=False, sharey=False,

figsize=(13, 8))

LabelsColors = []

# label的颜色配置矩阵

for i in datingLabels:

if i == 1:

LabelsColors.append("black")

if i == 2:

LabelsColors.append('orange')

if i == 3:

LabelsColors.append('red')

# 绘制散点图，以datingDataMat矩阵第一列为x，第二列为y，散点大小为15，透明度为0.5

axs[0][0].scatter(x=datingDataMat[:, 0], y=datingDataMat[:, 1],

color=LabelsColors, s=15, alpha=.5)

# 设置坐标轴的标目

axs0\_title\_text = axs[0][0].set\_title(u'每年获得的飞行常客里程数与玩视频游戏所消耗时间百分比', fontproperties=font)

axs0\_xlabel\_text = axs[0][0].set\_xlabel(u'每年获得的飞行常客里程数', fontproperties=font)

axs0\_ylabel\_text = axs[0][0].set\_ylabel(u'玩视频游戏所消耗时间百分比', fontproperties=font)

plt.setp(axs0\_title\_text, size=9, weight='bold', color='red')

plt.setp(axs0\_xlabel\_text, size=7, weight='bold', color='black')

plt.setp(axs0\_ylabel\_text, size=7, weight='bold', color='black')

# 绘制散点图，以datingDataMat矩阵第一列为x，第三列为y，散点大小为15，透明度为0.5

axs[0][1].scatter(x=datingDataMat[:, 0], y=datingDataMat[:, 2],

color=LabelsColors, s=15, alpha=.5)

axs1\_title\_text = axs[0][1].set\_title(u'每年获得的飞行常客里程数与每周消费的冰淇淋升数', fontproperties=font)

axs1\_xlabel\_text = axs[0][1].set\_xlabel(u'每年获得的飞行常客里程数', fontproperties=font)

axs1\_ylabel\_text = axs[0][1].set\_ylabel(u'每周消费的冰淇淋升数', fontproperties=font)

plt.setp(axs1\_title\_text, size=9, weight='bold', color='red')

plt.setp(axs1\_xlabel\_text, size=7, weight='bold', color='black')

plt.setp(axs1\_ylabel\_text, size=7, weight='bold', color='black')

# 绘制散点图，以datingDataMat矩阵第二列为x，第三列为y，散点大小为15，透明度为0.5

axs[1][0].scatter(x=datingDataMat[:, 1], y=datingDataMat[:, 2],

color=LabelsColors, s=15, alpha=.5)

axs2\_title\_text = axs[1][0].set\_title(u'玩视频游戏所消耗时间百分比与每周消费的冰淇淋升数', fontproperties=font)

axs2\_xlabel\_text = axs[1][0].set\_xlabel(u'玩视频游戏所消耗时间百分比', fontproperties=font)

axs2\_ylabel\_text = axs[1][0].set\_ylabel(u'每周消费的冰淇淋升数', fontproperties=font)

plt.setp(axs2\_title\_text, size=9, weight='bold', color='red')

plt.setp(axs2\_xlabel\_text, size=7, weight='bold', color='black')

plt.setp(axs2\_ylabel\_text, size=7, weight='bold', color='black')

# 设置图例

didntLike = mlines.Line2D([], [], color='black', marker='.', markersize=6,

label='didntLike')

smallDoses = mlines.Line2D([], [], color='orange', marker='.', markersize=6,

label='smallDoses')

largeDoses = mlines.Line2D([], [], color='red', marker='.', markersize=6,

label='largeDoses')

# 添加图例

axs[0][0].legend(handles=[didntLike, smallDoses, largeDoses])

axs[0][1].legend(handles=[didntLike, smallDoses, largeDoses])

axs[1][0].legend(handles=[didntLike, smallDoses, largeDoses])

# 显示图片

plt.show()

def autoNorm(dataSet):

# 对数据进行归一化

minVals = dataSet.min(0)

maxVals = dataSet.max(0)

ranges = maxVals - minVals

normDataSet = np.zeros(np.shape(dataSet))

m = dataSet.shape[0]

normDataSet = dataSet - np.tile(minVals, (m, 1))

# 原始值减去最小值（x - xmin）

normDataSet = normDataSet / np.tile(ranges, (m, 1))

# 上面的差值除以最大值和最小值之差

return normDataSet, ranges, minVals

def datingClassTest():

# 分类器测试函数

filename = "datingTestSet.txt"

# 将返回的特征矩阵和分类标签向量分别存储到datingDataMat和datingLabels中

datingDataMat, datingLabels = file2matrix(filename)

hoRatio = 0.10

# 取所有数据的10%，hoRatio越小，错误率越低

# 数据归一化，返回归一化数据结果、数据范围和最小值

normMat, ranges, minVals = autoNorm(datingDataMat)

m = normMat.shape[0]

numTestVecs = int(m \* hoRatio)

# 10%的测试数据的个数

errorCount = 0.0

# 分类错误计数

for i in range(numTestVecs):

# 前numTestVecs个数据作为测试集，后m-numTestVecs个数据作为训练集

# k选择标签数+1（结果比较好）

classifierResult = classify0(normMat[i, :], normMat[numTestVecs:m, :],

datingLabels[numTestVecs:m], 4)

print("分类结果：%d\t真实类别：%d" % (classifierResult, datingLabels[i]))

if classifierResult != datingLabels[i]:

errorCount += 1.0

print("错误率：%f%%" % (errorCount / float(numTestVecs) \* 100))

def classifyPerson():

# 输入一个人的3个特征，分类输出

resultList = ['讨厌', '有些喜欢', '非常喜欢']

percentTats = float(input("玩视频游戏所消耗时间百分比："))

ffMiles = float(input("每年获得的飞行常客里程数："))

iceCream = float(input("每周消费的冰淇淋升数："))

filename = "datingTestSet.txt"

datingDataMat, datingLabels = file2matrix(filename)

normMat, ranges, minVals = autoNorm(datingDataMat)

# 测试集归一化

inArr = np.array((percentTats, ffMiles, iceCream))

norminArr = (inArr - minVals) / ranges

# 分类并输出

classifierResult = classify0(norminArr, normMat, datingLabels, 4)

print("你可能%s这个人" % (resultList[classifierResult - 1]))

def main():

start = time.perf\_counter()

# 获取程序运行时间

filename = "datingTestSet.txt"

datingDataMat, datingLabels = file2matrix(filename)

normDataset, ranges, minVals = autoNorm(datingDataMat)

datingClassTest()

showdatas(datingDataMat, datingLabels)

classifyPerson()

end = time.perf\_counter()

print('Running time: %f seconds' % (end - start))

if \_\_name\_\_ == '\_\_main\_\_':

main()

datingTestSet.txt 数据集特征解释

