## **广州大学学生实验报告**

**开课学院及实验室：**计算机科学与工程实验室 **2021年10月30日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | **计算机科学与网络工程学院** | **年级/专业/班** | **网络193** | **姓名** | 洪泽彬 | **学号** | 1906200102 |
| **实验课程名称** | **人工智能导论实验** | | | | | **成绩** |  |
| **实验项目名称** | **分类算法** | | | | | **指导老师** | 张少宏 |

(\*\*\*报告只能为文字和图片,老师评语将添加到此处,学生请勿作答\*\*\*)

**一、实验内容**

**问题描述**：分类算法是解决分类问题的方法，是数据挖掘、机器学习和模式识别中一个重要的研究领域。分类算法通过对已知类别训练集的分析，从中发现分类规则，以此预测新数据的类别。分类算法的应用非常广泛，银行中风险评估、客户类别分类、文本检索和搜索引擎分类、安全领域中的入侵检测以及软件项目中的应用等等。

**内容提要**：针对教师指定的两类公用数据集（纯数值型例如UCIIris, 混杂型数据例如UCI Bank Marketing），学生至少实现两种分类算法，并比较分析结果原因。本次实验主要内容包括数据处理、算法实现和评价方法。鼓励与其他方法尤其是业界领先算法进行比较分析，鼓励创新设计求解方法。

**二、实验设备**

1. 实验设备：计算机；

2. 平台：Windows操作系统，Pycharm 2017

**三、实验步骤**

* 读取数据，并做预处理。
* 至少实现两种分类算法，选择评价方法比较结果并分析原因
* 选择适当可视化方法显示结果。
* 扩展选做题：分析考虑数据的特性并和具体分类方法的匹配。

**四、分析说明**

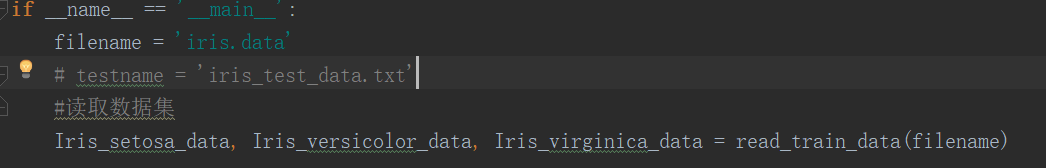
* 朴素贝叶斯分类算法

该算法的特点，使用概率统计的知识对样本数据集进行分类。给定一个属性值，其属于某个类的概率叫做条件概率。对于一个给定的类值，将每个属性的条件概率相乘，便得到一个数据样本属于某个类的概率。

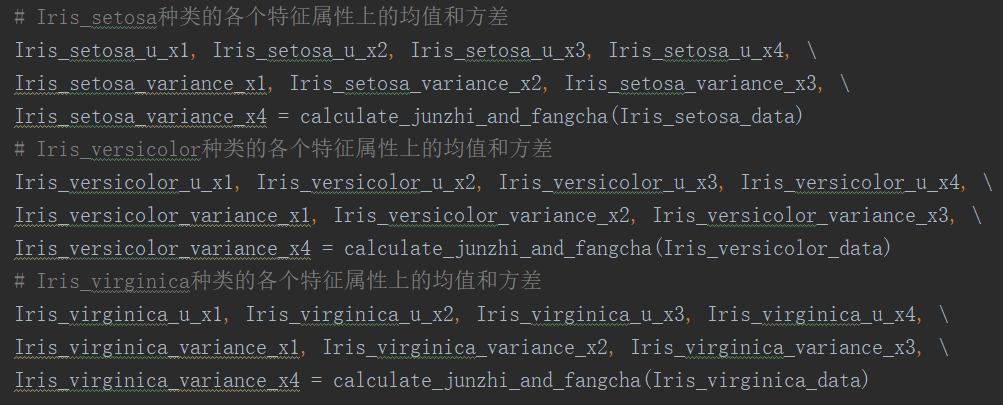
所需估计的参数很少，对缺失数据不太敏感，算法也比较简单。

因此，使用改算法对iris.data数据集进行分类。

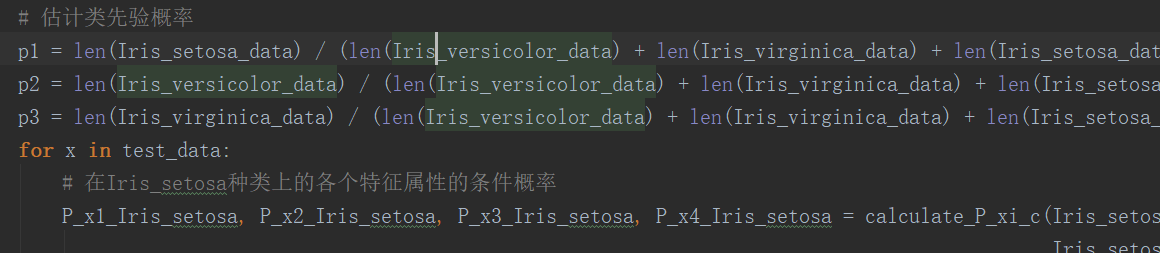
读取数据，对数据进行预处理，这里我将每种花取前45条数据，剩下的5条数据作为测试数据集。



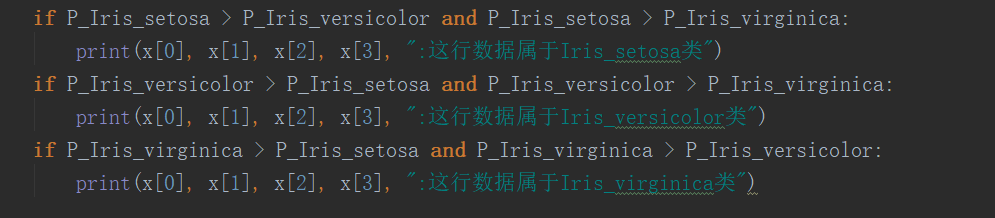
处理数据，求取均值和方差。



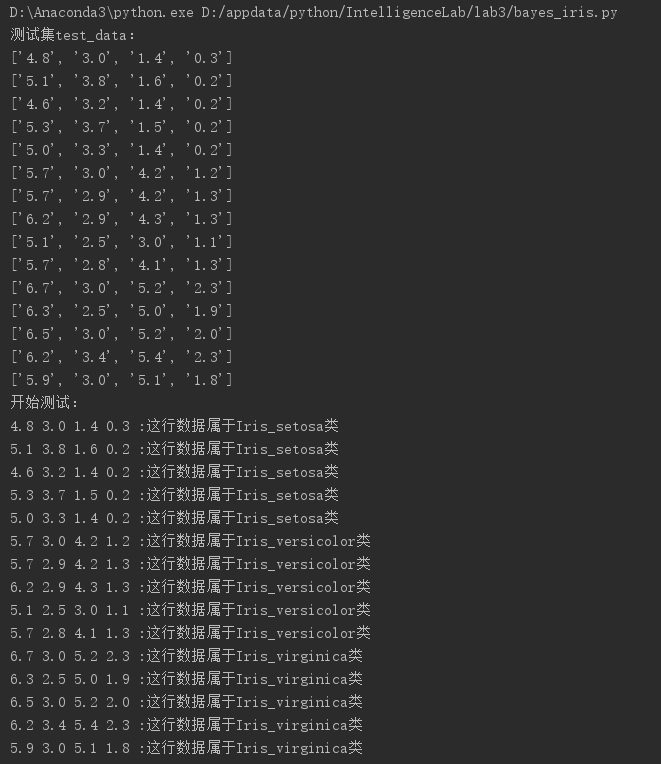
首先估计先验概率，这里我每个类别所占整体数据集的比例是一样的，利用概率密度函数，计算测试数据集上各个属性在每个类别上的条件概率，计算后验概率=先验概率\*条件概率，比较在各个类别上的后验概率，取最大值，则分为这个类别



最后对结果进行判断，并进行可视化：



结果：

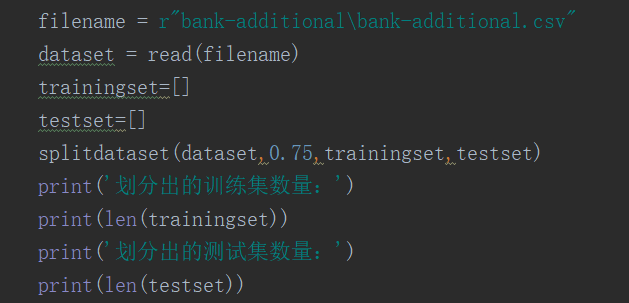


我们将结果与测试集比较发现结果完全正确！

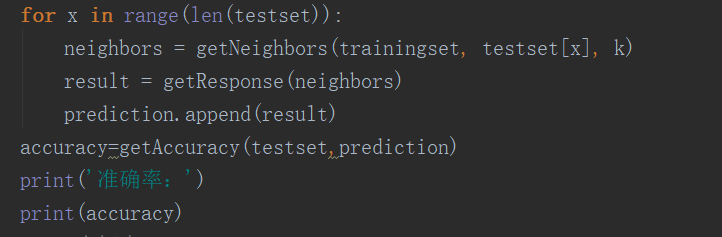
* Knn算法

Knn算法的主要思路是：如果一个样本在特征空间中的K个最相似（即特征空间中最邻近）的样本中的大多数属于某一个类别，则该样本也属于这个类别。这是一个将数据集合中每一个记录进行分类的方法。所以可对记录特征多的数据集进行分类，因此这里对bank-additional数据集进行分类。

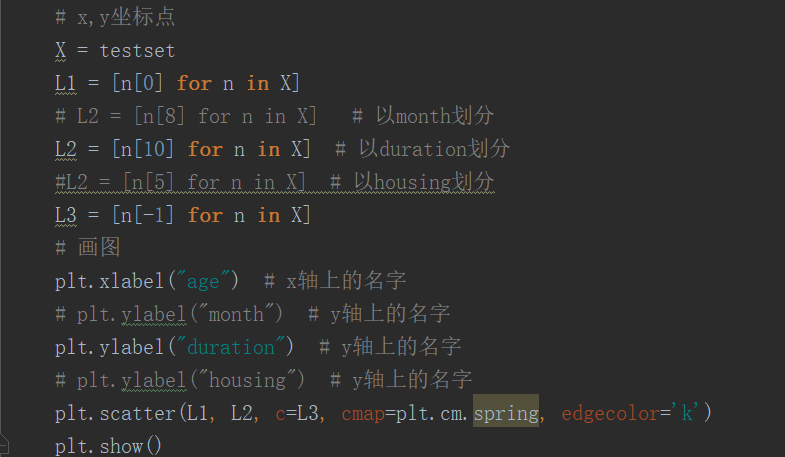
数据预处理和对数据进行划分，分出测试数据集：



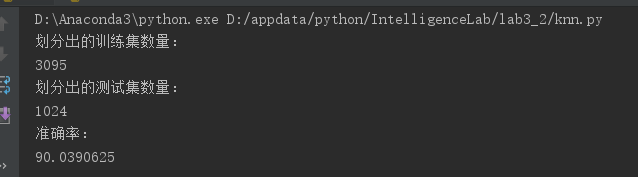
计算处欧氏距离，找出距离最近的k个实例，获取距离最近的k个实例中占比例较大的分类：



可视化：

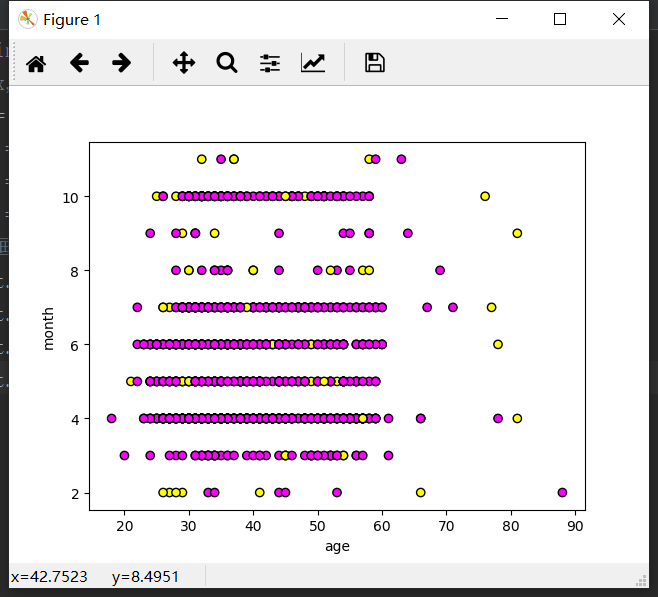


结果：

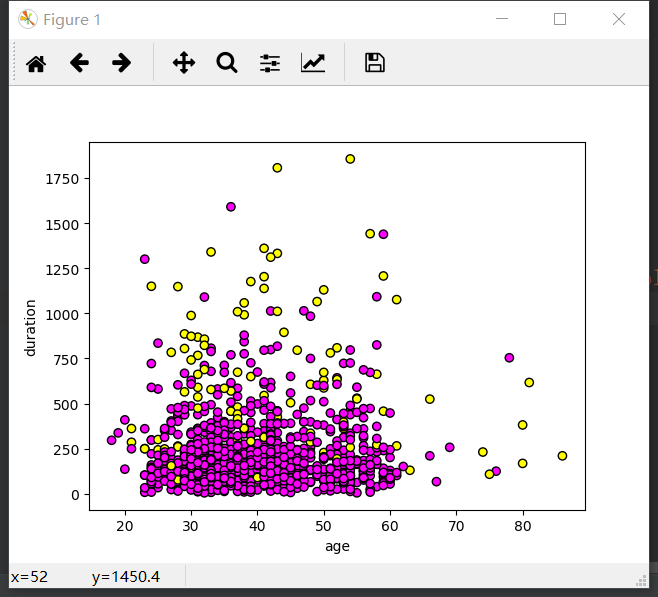


可视化绘制分类结果图像（这里仅以month，duration，housing作为展示，其余字段修改参数即可得到图像）：

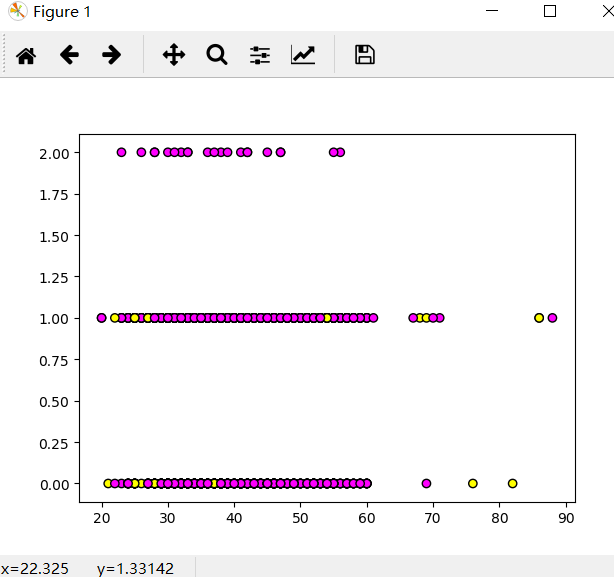
month字段的划分结果：



duration字段的划分结果：



以housing字段划分结果：

****

**实验总结：**

通过本次实验，我了解并熟悉分类算法的原理以及使用，并通过分类算法分类了数据集iris\_data，bank-additional，并体会到了各种分类算法的思想以及优缺点。如knn算法，每个样别都可以用它k个最邻近的值来代表。同时，该算法距离也用到欧氏距离进行求解，和上一次实验又有联系，让我觉得每做一个实验就是在筑起一个地基。总而言之，希望能在今后的实验中学到更多有用的知识。

**函数源代码如下：**

**bayes\_iris.py:**

# coding:utf-8

import math

Iris\_setosa\_data = []

Iris\_versicolor\_data = []

Iris\_virginica\_data = []

test\_data = []

# 读取训练数据集，这里我将每种花取前45条数据，剩下的5条数据作为测试数据集

def read\_train\_data(filename):

f = open(filename, 'r')

all\_lines = f.readlines()

for line in all\_lines[0:45]: #[0:45]左闭右开

line = line.strip().split(',')

Iris\_setosa\_data.append(line[0:4])

# Iris\_setosa\_label+=1

for line in all\_lines[45:50]:

line = line.strip().split(',')

test\_data.append(line[0:4])

for line in all\_lines[50:95]:

line = line.strip().split(',')

Iris\_versicolor\_data.append(line[0:4])

# Iris\_versicolor\_label+=1

for line in all\_lines[95:100]:

line = line.strip().split(',')

test\_data.append(line[0:4])

for line in all\_lines[100:145]:

line = line.strip().split(',')

Iris\_virginica\_data.append(line[0:4])

# Iris\_virginica\_label+=1

for line in all\_lines[145:150]:

line = line.strip().split(',')

test\_data.append(line[0:4])

return Iris\_setosa\_data, Iris\_versicolor\_data, Iris\_virginica\_data

# 读取测试数据集

# def read\_test\_data(testname):

# f = open(testname, 'r')

# all\_lines = f.readlines()

# for line in all\_lines[0:]:

# line = line.strip().split(',') # 以逗号为分割符拆分列表

# test\_data.append(line)

# return test\_data

# 计算均值和方差

def calculate\_junzhi\_and\_fangcha(train\_data):

x1\_sum = 0.0

x2\_sum = 0.0

x3\_sum = 0.0

x4\_sum = 0.0

for x in train\_data: # 计算各个特征的和

x1\_sum += float(x[0])

x2\_sum += float(x[1])

x3\_sum += float(x[2])

x4\_sum += float(x[3])

# print(x[0],x[1],x[2],x[3])

# 计算样本在各个属性上取值的均值

u\_x1 = x1\_sum / 45

u\_x2 = x2\_sum / 45

u\_x3 = x3\_sum / 45

u\_x4 = x4\_sum / 45

k1 = 0.0

k2 = 0.0

k3 = 0.0

k4 = 0.0

# 计算各类样本在第i个属性上的方差

for x in train\_data:

k1 += (float(x[0]) - u\_x1) \*\* 2

k2 += (float(x[1]) - u\_x2) \*\* 2

k3 += (float(x[2]) - u\_x3) \*\* 2

k4 += (float(x[3]) - u\_x4) \*\* 2

variance\_x1 = k1 / 45

variance\_x2 = k2 / 45

variance\_x3 = k3 / 45

variance\_x4 = k4 / 45

return u\_x1, u\_x2, u\_x3, u\_x4, variance\_x1, variance\_x2, variance\_x3, variance\_x4

# 计算每个属性估计条件概率

def calculate\_P\_xi\_c(u\_x1, u\_x2, u\_x3, u\_x4, variance\_x1, variance\_x2, variance\_x3, variance\_x4, line\_data):

p\_x1\_c = (1 / math.sqrt(2 \* math.pi)) \* math.exp(-(float(line\_data[0]) - u\_x1) \*\* 2 / (2 \* variance\_x1))

p\_x2\_c = (1 / math.sqrt(2 \* math.pi)) \* math.exp(-(float(line\_data[1]) - u\_x2) \*\* 2 / (2 \* variance\_x2))

p\_x3\_c = (1 / math.sqrt(2 \* math.pi)) \* math.exp(-(float(line\_data[2]) - u\_x3) \*\* 2 / (2 \* variance\_x3))

p\_x4\_c = (1 / math.sqrt(2 \* math.pi)) \* math.exp(-(float(line\_data[3]) - u\_x4) \*\* 2 / (2 \* variance\_x4))

return p\_x1\_c, p\_x2\_c, p\_x3\_c, p\_x4\_c

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

filename = 'iris.data'

# testname = 'iris\_test\_data.txt'

#读取数据集

Iris\_setosa\_data, Iris\_versicolor\_data, Iris\_virginica\_data = read\_train\_data(filename)

# Iris\_setosa种类的各个特征属性上的均值和方差

Iris\_setosa\_u\_x1, Iris\_setosa\_u\_x2, Iris\_setosa\_u\_x3, Iris\_setosa\_u\_x4, \

Iris\_setosa\_variance\_x1, Iris\_setosa\_variance\_x2, Iris\_setosa\_variance\_x3, \

Iris\_setosa\_variance\_x4 = calculate\_junzhi\_and\_fangcha(Iris\_setosa\_data)

# Iris\_versicolor种类的各个特征属性上的均值和方差

Iris\_versicolor\_u\_x1, Iris\_versicolor\_u\_x2, Iris\_versicolor\_u\_x3, Iris\_versicolor\_u\_x4, \

Iris\_versicolor\_variance\_x1, Iris\_versicolor\_variance\_x2, Iris\_versicolor\_variance\_x3, \

Iris\_versicolor\_variance\_x4 = calculate\_junzhi\_and\_fangcha(Iris\_versicolor\_data)

# Iris\_virginica种类的各个特征属性上的均值和方差

Iris\_virginica\_u\_x1, Iris\_virginica\_u\_x2, Iris\_virginica\_u\_x3, Iris\_virginica\_u\_x4, \

Iris\_virginica\_variance\_x1, Iris\_virginica\_variance\_x2, Iris\_virginica\_variance\_x3, \

Iris\_virginica\_variance\_x4 = calculate\_junzhi\_and\_fangcha(Iris\_virginica\_data)

'''开始测试'''

# test\_data = read\_test\_data(testname)

print ('测试集test\_data：')

for x in test\_data:

print(x)

print ("开始测试：")

# 估计类先验概率

p1 = len(Iris\_setosa\_data) / (len(Iris\_versicolor\_data) + len(Iris\_virginica\_data) + len(Iris\_setosa\_data))

p2 = len(Iris\_versicolor\_data) / (len(Iris\_versicolor\_data) + len(Iris\_virginica\_data) + len(Iris\_setosa\_data))

p3 = len(Iris\_virginica\_data) / (len(Iris\_versicolor\_data) + len(Iris\_virginica\_data) + len(Iris\_setosa\_data))

for x in test\_data:

# 在Iris\_setosa种类上的各个特征属性的条件概率

P\_x1\_Iris\_setosa, P\_x2\_Iris\_setosa, P\_x3\_Iris\_setosa, P\_x4\_Iris\_setosa = calculate\_P\_xi\_c(Iris\_setosa\_u\_x1,

Iris\_setosa\_u\_x2,

Iris\_setosa\_u\_x3,

Iris\_setosa\_u\_x4, \

Iris\_setosa\_variance\_x1,

Iris\_setosa\_variance\_x2,

Iris\_setosa\_variance\_x3,

Iris\_setosa\_variance\_x4,

x)

# print(P\_x1\_Iris\_setosa,P\_x2\_Iris\_setosa,P\_x3\_Iris\_setosa,P\_x4\_Iris\_setosa)

# 在Iris\_versicolor种类上的各个特征属性的条件概率

P\_x1\_Iris\_versicolor, P\_x2\_Iris\_versicolor, P\_x3\_Iris\_versicolor, P\_x4\_Iris\_versicolor = calculate\_P\_xi\_c(

Iris\_versicolor\_u\_x1, Iris\_versicolor\_u\_x2, Iris\_versicolor\_u\_x3, Iris\_versicolor\_u\_x4, \

Iris\_versicolor\_variance\_x1, Iris\_versicolor\_variance\_x2, Iris\_versicolor\_variance\_x3,

Iris\_versicolor\_variance\_x4, x)

# print(P\_x1\_Iris\_versicolor,P\_x2\_Iris\_versicolor,P\_x3\_Iris\_versicolor)

# 在Iris\_virginica种类上的各个特征属性的条件概率

P\_x1\_Iris\_virginica, P\_x2\_Iris\_virginica, P\_x3\_Iris\_virginica, P\_x4\_Iris\_virginica = calculate\_P\_xi\_c(

Iris\_virginica\_u\_x1, Iris\_virginica\_u\_x2, Iris\_virginica\_u\_x3, Iris\_virginica\_u\_x4, \

Iris\_virginica\_variance\_x1, Iris\_virginica\_variance\_x2, Iris\_virginica\_variance\_x3,

Iris\_virginica\_variance\_x4, x)

# print(P\_x1\_Iris\_virginica,P\_x2\_Iris\_virginica,P\_x3\_Iris\_virginica,P\_x4\_Iris\_virginica)

# 计算各个种类上的后验概率

P\_Iris\_setosa = p1 \* P\_x1\_Iris\_setosa \* P\_x2\_Iris\_setosa \* P\_x3\_Iris\_setosa \* P\_x4\_Iris\_setosa

# print( P\_Iris\_setosa)

P\_Iris\_versicolor = p2 \* P\_x1\_Iris\_versicolor \* P\_x2\_Iris\_versicolor \* P\_x3\_Iris\_versicolor \* P\_x4\_Iris\_versicolor

# print( P\_Iris\_versicolor)

P\_Iris\_virginica = p3 \* P\_x1\_Iris\_virginica \* P\_x2\_Iris\_virginica \* P\_x3\_Iris\_virginica \* P\_x4\_Iris\_virginica

# print( P\_Iris\_virginica)

if P\_Iris\_setosa > P\_Iris\_versicolor and P\_Iris\_setosa > P\_Iris\_virginica:

print(x[0], x[1], x[2], x[3], ":这行数据属于Iris\_setosa类")

if P\_Iris\_versicolor > P\_Iris\_setosa and P\_Iris\_versicolor > P\_Iris\_virginica:

print(x[0], x[1], x[2], x[3], ":这行数据属于Iris\_versicolor类")

if P\_Iris\_virginica > P\_Iris\_setosa and P\_Iris\_virginica > P\_Iris\_versicolor:

print(x[0], x[1], x[2], x[3], ":这行数据属于Iris\_virginica类")

**knn.py:**

import pandas as pd

import random

import math

import operator

import matplotlib.pyplot as plt

# 转换职业状态

# "admin.","blue-collar","entrepreneur","housemaid","management","retired","self-employed","services","student","technician","unemployed","unknown"

def change\_job(state):

job = [ "admin.","blue-collar","entrepreneur","housemaid","management","retired",

"self-employed","services","student","technician","unemployed","unknown"]

for i in range(len(job)):

if state == job[i]:

return i

# 转换婚姻状态

# 未知：0、单身：1、结婚：2、离婚：3

def change\_marital(state):

marital = ["divorced","married","single","unknown"]

for i in range(len(marital)):

if state == marital[i]:

return i

# 转换教育水平

# "basic.4y","basic.6y","basic.9y","high.school","illiterate","professional.course","university.degree","unknown"

def change\_education(state):

education = ["basic.4y","basic.6y","basic.9y","high.school","illiterate","professional.course","university.degree","unknown"]

for i in range(len(education)):

if state == education[i]:

return i

# 二分类转换

# yes：1、no：0

def change\_binary(state):

if state == 'yes':

state = 1

elif state == 'no':

state = 0

return state

# 转换接触方式

# 未知(unknown):0、网络(cellular)：1、电话（telephone）：2

def change\_contact(state):

contact = ['unknown', 'cellular', 'telephone']

for i in range(len(contact)):

if state == contact[i]:

return i

# 转换poutcome（上次成功与否）

# "failure","nonexistent","success"

def change\_poutcome(state):

poutcome = ["failure","nonexistent","success"]

for i in range(len(poutcome)):

if state == poutcome[i]:

return i

# 转换month

def change\_month(state):

month = ['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec']

for i in range(len(month)):

if state == month[i]:

return i

# 转换week

def change\_week(state):

week = ["mon","tue","wed","thu","fri"]

for i in range(len(week)):

if state == week[i]:

return i

# 转换default

def change\_default(state):

default = ["no","yes","unknown"]

for i in range(len(default)):

if state == default[i]:

return i

def read(filename):

# 读取csv文件数据集

bank = pd.read\_csv(filename, sep=';')

# 创建list

data = [[] for i in range(len(bank))]

# 逐个数据处理

for i in range(len(data)):

data[i].append(bank['age'][i])

data[i].append(change\_job(bank['job'][i]))

data[i].append(change\_marital(bank['marital'][i]))

data[i].append(change\_education(bank['education'][i]))

data[i].append(change\_default(bank['default'][i]))

data[i].append(change\_default(bank['housing'][i]))

data[i].append(change\_default(bank['loan'][i]))

data[i].append(change\_contact(bank['contact'][i]))

data[i].append(change\_month(bank['month'][i]))

data[i].append(change\_week(bank['day\_of\_week'][i]))

data[i].append(bank['duration'][i])

data[i].append(bank['campaign'][i])

data[i].append(bank['pdays'][i])

data[i].append(bank['previous'][i])

data[i].append(change\_poutcome(bank['poutcome'][i]))

data[i].append(bank['emp.var.rate'][i])

data[i].append(bank['cons.price.idx'][i])

data[i].append(bank['cons.conf.idx'][i])

data[i].append(bank['euribor3m'][i])

data[i].append(bank['nr.employed'][i])

data[i].append(change\_binary(bank['y'][i]))

#把数据转为浮点数类型

for x in range(len(data)):

for y in range(len(data[x])):

# print(data[x][y])

data[x][y] = float(data[x][y])

return data

#拆分数据集 dataset(要拆分的数据集) split(训练集所占比例) testset(测试集) trainingset(训练集)

def splitdataset(dataset,split,trainingset,testset):

for x in range(len(dataset)):

if random.random()<=split:

trainingset.append(dataset[x])

else:

testset.append(dataset[x])

#计算欧式距离 instance1(第一个坐标点) instance2(第二个坐标点) length(特征值个数)

def euclideanDistance(instance1,instance2,length):

distance=0

for x in range(length):

distance += pow((instance1[x] - instance2[x]), 2)

return math.sqrt(distance)

#选取距离最近的k个实例 testinstance(需要分类的测试集点)

def getNeighbors(trainingset,testinstance,k):

distances=[]

length=len(testinstance)-1

for x in range(len(trainingset)):

dis=euclideanDistance(trainingset[x],testinstance,length)

distances.append((trainingset[x],dis))

distances.sort(key=operator.itemgetter(1))

neighbors=[]

for x in range(k):

neighbors.append(distances[x][0])

return neighbors

#获取距离最近的k个实例中占比例较大的分类

def getResponse(neighbors):

classvotes={}

for x in range(len(neighbors)):

response=neighbors[x][-1]

if response in classvotes:

classvotes[response]+=1

else:

classvotes[response]=1

sortedVotes = sorted(classvotes.items(), key=operator.itemgetter(1), reverse=True)

return sortedVotes[0][0]

#计算准确率

def getAccuracy(testset,prediction):

correct=0

for x in range(len(testset)):

if testset[x][-1]==prediction[x]:

correct+=1

return correct/float(len(testset))\*100.0

def main():

filename = r"bank-additional\bank-additional.csv"

dataset = read(filename)

trainingset=[]

testset=[]

splitdataset(dataset,0.75,trainingset,testset)

print('划分出的训练集数量：')

print(len(trainingset))

print('划分出的测试集数量：')

print(len(testset))

prediction=[]

k=7

for x in range(len(testset)):

neighbors = getNeighbors(trainingset, testset[x], k)

result = getResponse(neighbors)

prediction.append(result)

accuracy=getAccuracy(testset,prediction)

print('准确率：')

print(accuracy)

# x,y坐标点

X = testset

L1 = [n[0] for n in X]

# L2 = [n[8] for n in X] # 以month划分

L2 = [n[10] for n in X] # 以duration划分

#L2 = [n[5] for n in X] # 以housing划分

L3 = [n[-1] for n in X]

# 画图

plt.xlabel("age") # x轴上的名字

# plt.ylabel("month") # y轴上的名字

plt.ylabel("duration") # y轴上的名字

# plt.ylabel("housing") # y轴上的名字

plt.scatter(L1, L2, c=L3, cmap=plt.cm.spring, edgecolor='k')

plt.show()

main()