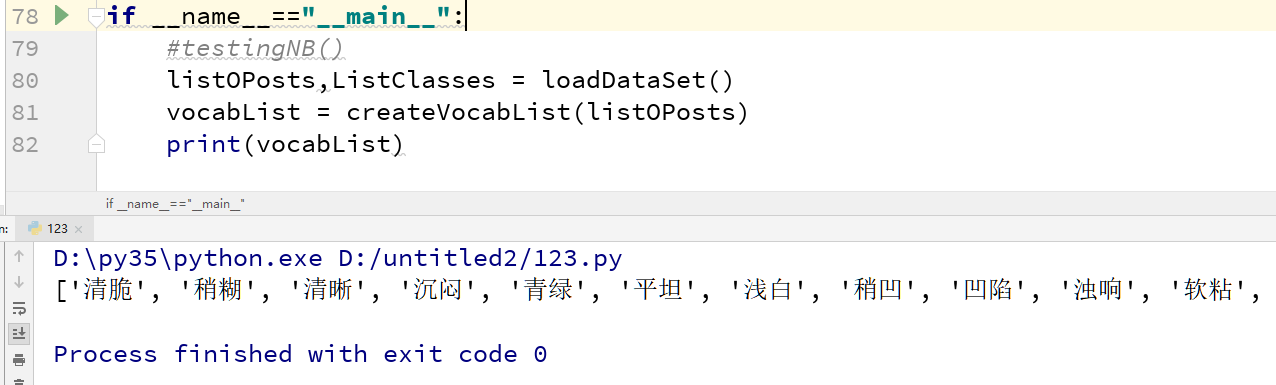
作业五

Zs2016061002002 戚宏成

**1 以西瓜数据集2.0（见教材76页表4.1）中样本1--16为训练集训练一个朴素贝叶斯分类器，对测试样本17进行分类。请写出详细的计算过程。**

1. **根据所有的训练样本创建一个所有属性集合,并获取所有不重复属性的集合如下图:**



1. **根据(1)中所有不重复的属性集合对每个属性内容向量化,如下图**

listOPosts,ListClasses = loadDataSet()

vocabList = createVocabList(listOPosts)

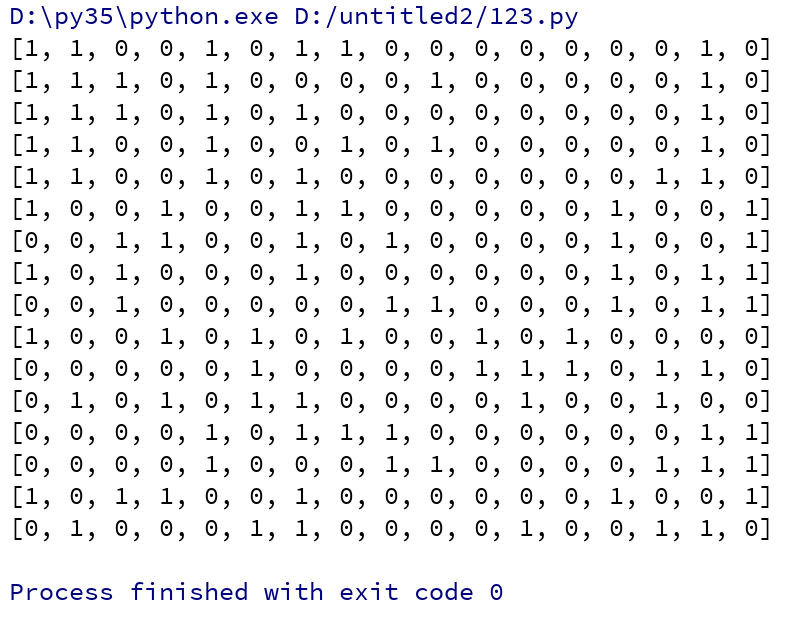
trainMat=[]

for postinDoc in listOPosts:

trainMat.append(bagOfWords2VecMN(vocabList,postinDoc))

for i in trainMat:

print(i)



1. **训练模型，此时就是分别求p(坏瓜|好瓜) = p(坏瓜)\*p（好瓜|坏瓜）/p(好瓜):**

listOPosts,ListClasses = loadDataSet()

vocabList = createVocabList(listOPosts)

trainMat=[]

for postinDoc in listOPosts:

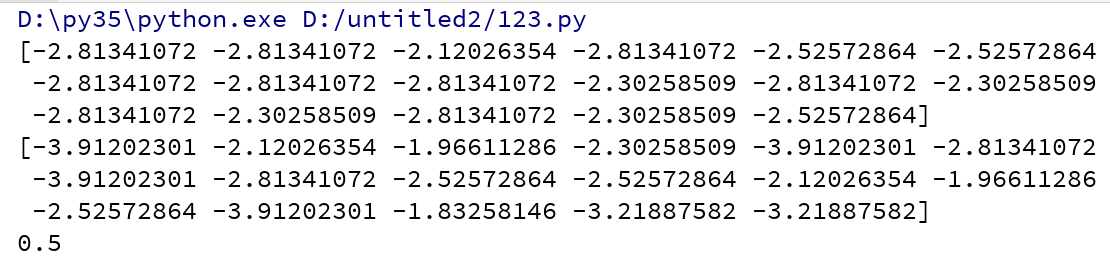
trainMat.append(bagOfWords2VecMN(vocabList,postinDoc))

p0V ,p1V,pAV=trainNBO(trainMat,ListClasses)

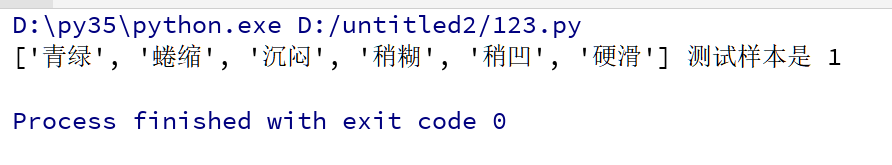
print(p0V)

print(p1V)

print(pAV)



1. **定义分类方法进行测试:**



**代码如下:**

import numpy

import math

def loadDataSet():

postingList =[['青绿', '蜷缩', '浊响', '清晰', '凹陷', '硬滑'],#1

['乌黑', '蜷缩', '沉闷', '清晰', '凹陷', '硬滑'],#2

['乌黑', '蜷缩', '浊响', '清晰', '凹陷', '硬滑'],#3

['青绿', '蜷缩', '沉闷', '清晰', '凹陷', '硬滑'],#4

['浅白', '蜷缩', '浊响', '清晰', '凹陷', '硬滑'],#5

['青绿', '稍蜷', '浊响', '清晰', '稍凹', '软粘'],#6

['乌黑', '稍蜷', '浊响', '稍糊', '稍凹', '软粘'],#7

['乌黑', '稍蜷', '浊响', '清晰', '稍凹', '硬滑'],#8

#############################################

['乌黑', '稍蜷', '沉闷', '稍糊', '稍凹', '硬滑'],#9

['青绿', '硬挺', '清脆', '清晰', '平坦', '软粘'],#10

['浅白', '硬挺', '清脆', '模糊', '平坦', '硬滑'],#11

['浅白', '蜷缩', '浊响', '模糊', '平坦', '软粘'],#12

['青绿', '稍蜷', '浊响', '稍糊', '凹陷', '硬滑'],#13

['浅白', '稍蜷', '沉闷', '稍糊', '凹陷', '硬滑'],#14

['乌黑', '稍蜷', '浊响', '清晰', '稍凹', '软粘'],#15

['浅白', '蜷缩', '浊响', '模糊', '平坦', '硬滑']#16

]

classVec =[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0]

return postingList,classVec

def createVocabList(dataSet):

vocabSet =set([])

for document in dataSet:

vocabSet = vocabSet | set(document)

return list(vocabSet)

def bagOfWords2VecMN(vocabList,inputSet):

returnVec =[0]\*len(vocabList)

for word in inputSet:

returnVec[vocabList.index(word)] +=1

return returnVec

def trainNBO(trainMatrix,trainCategory):

numTrainDocs = len(trainMatrix)

numWords =len(trainMatrix[0])

#计算p(坏瓜)的概率

pAbusive = sum(trainCategory)/float(numTrainDocs)

#为了防止一个概率为0，假设都有一个

p0Num =numpy.ones(numWords)

p1Num =numpy.ones(numWords)

p0Denom =2.0;p1Denom=2.0

for i in range(numTrainDocs):

if trainCategory[i] ==1:

p1Num +=trainMatrix[i]

p1Denom +=sum(trainMatrix[i])

else:

p0Num +=trainMatrix[i]

p0Denom +=sum(trainMatrix[i])

p1Vect = numpy.log((p1Num/p1Denom))

p0Vect = numpy.log(p0Num/p0Denom)

return p0Vect,p1Vect,pAbusive

def classifyNB(vec2Classify,p0Vec,p1Vec,pClass1):

p1 =sum(vec2Classify \* p1Vec) +math.log(pClass1)

p0 = sum(vec2Classify \* p0Vec)+math.log(1.0-pClass1)

if p1>p0:

return 1

else:

return 0

def testingNB():

listOPosts,ListClasses = loadDataSet();

myVocabList = createVocabList(listOPosts)

trainMat=[]

for postinDoc in listOPosts:

trainMat.append(bagOfWords2VecMN(myVocabList,postinDoc))

p0V,p1V,pAb =trainNBO(trainMat,ListClasses)

testEntry =['青绿','蜷缩','沉闷','稍糊','稍凹','硬滑']

######testEntry =['浅白', '蜷缩', '浊响', '模糊', '平坦', '硬滑'] --🡪0

thisDoc = numpy.array(bagOfWords2VecMN(myVocabList,testEntry))

print(testEntry,'测试样本是',classifyNB(thisDoc,p0V,p1V,pAb) )

if \_\_name\_\_=="\_\_main\_\_":

testingNB()