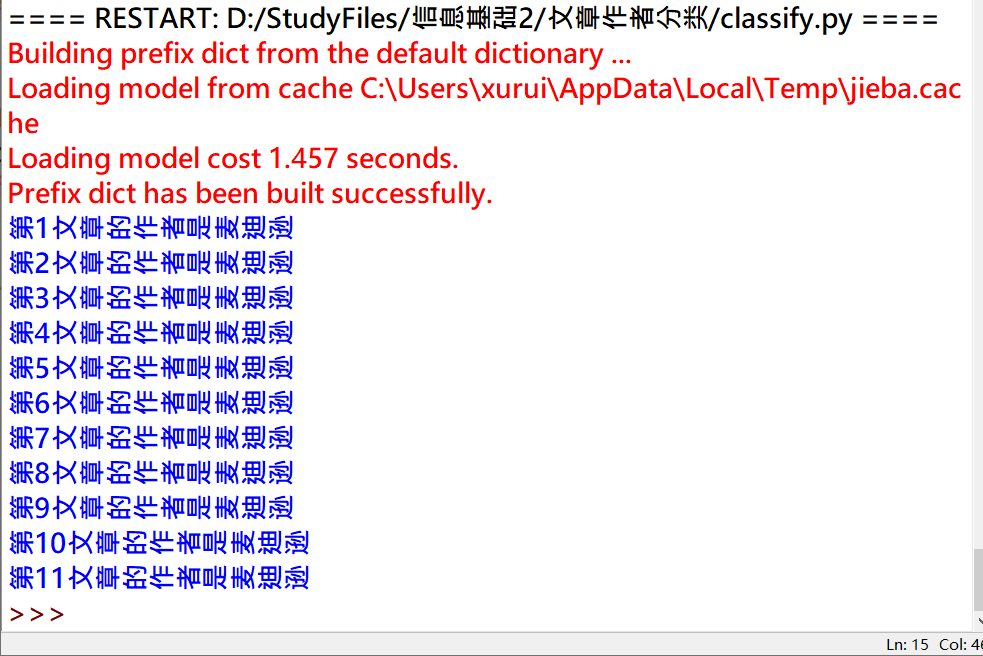
信息基础2 作业

徐瑞 20180012045

文章分类：**HAMILTON OR MADISON**

85篇文章中，有11篇（49-57，62，63）是作者不确定的HAMILTON OR MADISON，需要进行分类，要根据其它的已经明确作者的文章来判断，剔除作者JAY的文章（2，3，4，5，64）后明确的HAMILTON和MADISON的文章共有69篇，我的思路是先通过分析每篇文章中关键词（根据权重判断）和作者来判断这两个作者的文章的特点，再统计一下未知作者的11篇文章的关键词，通过和已知文章中关键词的比较就可以完成文章作者的分类预测。

这个题目需要先进行一个数据预处理，我把已知作者是汉密尔顿的全部文章保存为A\_article.txt，把已知作者是麦迪逊的全部文章保存为B\_article.txt，然后再把要分类的分别文章保存为1\_article.txt，2\_article.txt，3\_article.txt，4\_article.txt，5\_article.txt，6\_article.txt，7\_article.txt，8\_article.txt，9\_article.txt，10\_article.txt，11\_article.txt，然后再分别进行关键词提取和分类，对这11篇文章的分类结果如下图所示：



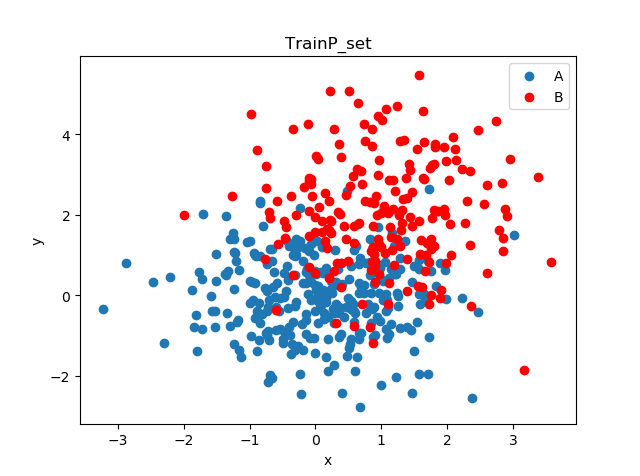
详细代码如下：

from jieba.analyse import \*  
  
A\_list = []  
B\_list = [] #分别存放A和B的关键词  
with open(**'A\_article.txt'**) as f:  
 data = f.read()  
 for A\_keyword, A\_weight in extract\_tags(data,topK=100, withWeight=True):  
 A\_list.append(A\_keyword) #A类关键词  
 #print('%s %s' % (A\_keyword,A\_weight)) #打印出A类文章权重前100的关键词和权重  
with open(**'B\_article.txt'**) as f:  
 data = f.read()  
 for B\_keyword, B\_weight in extract\_tags(data, topK=100,withWeight=True):  
 B\_list.append(B\_keyword) #B类关键词  
 #print('%s %s' % (B\_keyword, B\_weight)) #打印出A类文章权重前100的关键词和权重  
def classify(x):  
 with open(str(x)+**'\_article.txt'**) as f:  
 data = f.read()  
 a=0;  
 b=0;  
 for keyword, weight in extract\_tags(data, topK=100,withWeight=True):  
 if keyword in A\_list:  
 a += 1  
 else:  
 b += 1  
 if a>b:  
 print(**'第'**+str(x)+**'文章的作者是汉密尔顿'**)  
 else:  
 print(**'第'**+str(x)+**'文章的作者是麦迪逊'**)  
for x in range(1,12):  
 classify(x)

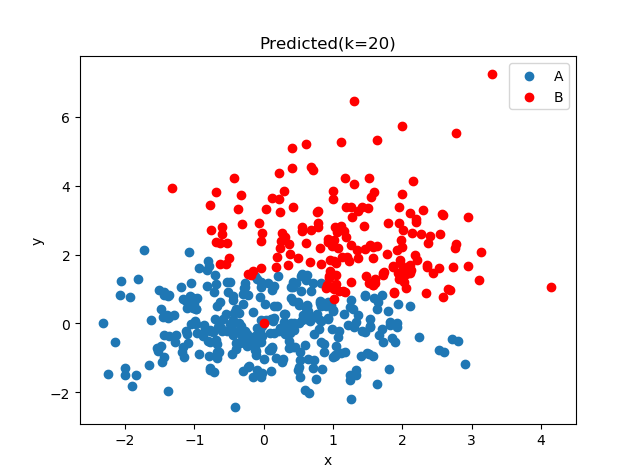
KNN高斯分布

问题：有两类空间点，A类和B类。A 类点以(0, 0)为中心、(1, 0; 0, 1)为协方差矩阵的二维高斯分布；B 类点以(1, 2) 为中心、(1, 0; 0, 2)为协方差矩阵的二维高斯分布；随机生成300个A类点，200个B类点，并用k-最近邻的方法进行分类。

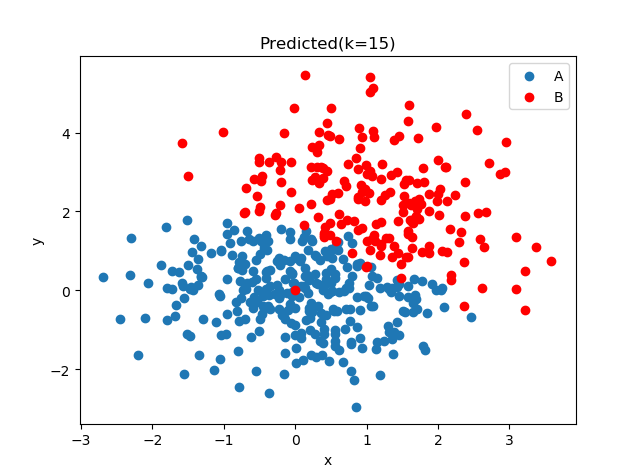
答案：我先通过numpy随机生成300个A类点和200个B类点，然后通过matplotlib将它们画出来，蓝色点代表A类点，红色点代表B类点，如下图所示：



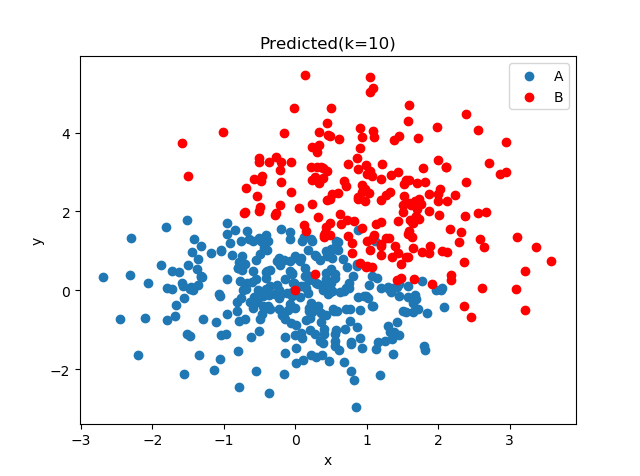
然后我构建了TrainP\_set和Train\_lable分别来装载A,B类点的数据和它们的标签，接着我又重新随机生成了300个A类点和200个B类点作为后面的测试集，之后就开始对测试集进行KNN分类，没有使用相关库函数，自己写的，距离采用的是欧氏距离，利用collections 中Counter函数的投票机制完成分类，根据分类后的标签完成对应数据的分类，然后再画出分类后的效果图，还是蓝色点代表A类点，红色点代表B类点，我分别尝试了不同的K值，相应结果如下面所示：



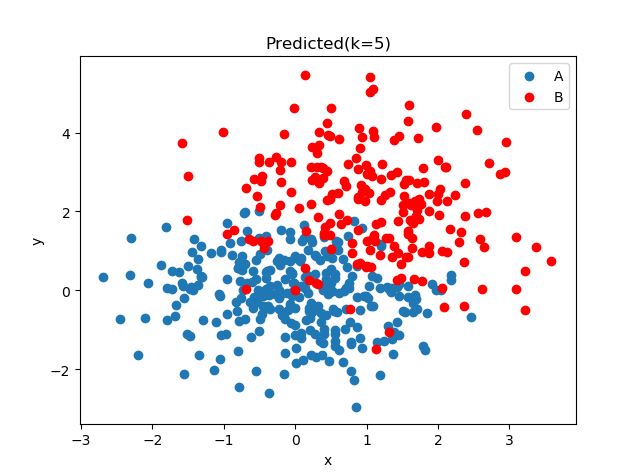
K=20时分类正确率为0.856；



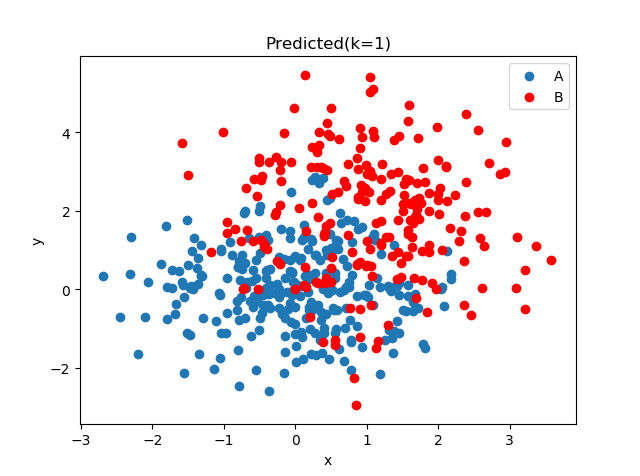
k=15时 分类正确率为： 0.84；



k=10时 分类正确率为： 0.838；



k=5时 分类正确率为： 0.8240000000000001；



k=1时 分类正确率为： 0.764；

每次随机产生的点不相同，所以分类后的正确率也不同，如果选择较小的k值，相当于用较小的领域中的训练实例进行预测，“学习”近似误差会减小，只有与输入实例较近或相似的训练实例才会对预测结果起作用，这样“学习”的估计误差会增大，容易发生过拟合；如果选择较大的k值，就相当于用较大领域中的训练实例进行预测与输入实例较远的训练实例也会对预测起作用，使预测发生错误，且k值的增大就意味着整体的模型变得简单，容易欠拟合。

详细代码如下：

#!/usr/bin/env python  
# coding: utf-8  
#18崇新 徐瑞 201800121045  
import numpy as np  
import matplotlib.pyplot as plt  
from collections import Counter  
  
  
mean1 = [0,0]  
cov1 = [[1,0],[0,1]]  
P\_set1 = np.random.multivariate\_normal(mean1,cov1,300)  
label\_A = np.array([**'A'**]\*300)  
  
  
mean2 = [1,2]  
cov2 = [[1,0],[0,2]]  
P\_set2 = np.random.multivariate\_normal(mean2,cov2,200)  
label\_B = np.array([**'B'**]\*200)  
  
  
x1,y1 = P\_set1.T  
x2,y2 = P\_set2.T  
plt.axis()  
plt.title(**"TrainP\_set"**)  
plt.xlabel(**"x"**)  
plt.ylabel(**"y"**)  
plt.scatter(x1,y1,label = **'A'**)  
plt.scatter(x2,y2,c = **'r'**,label = **'B'**)  
plt.legend()  
plt.show()  
  
  
TrainP\_set = np.append(P\_set1,P\_set2,axis=0)  
Train\_label = np.append(label\_A,label\_B)  
#重新随机生成一组满足题意的随机点作为测试集  
TestP\_set = np.append(np.random.multivariate\_normal(mean1,cov1,300),np.random.multivariate\_normal(mean2,cov2,200),axis=0)  
Test\_label = Train\_label  
**'''  
print(TrainP\_set.shape)  
print(Train\_lable.shape)  
'''**# 下面进行KNN分类  
def KNN(k,TrainP\_set,TestP\_set):  
 Sort\_label = [] #分类后的标签初始化  
 for i in range(500):  
 x,y = TestP\_set[i]  
 length = []  
 np.array(length)  
 for j in range(500):  
 m = (x-TrainP\_set[j][0])\*\*2  
 n = (y-TrainP\_set[j][1])\*\*2  
 length = np.append(length,np.sqrt(m+n)) #距离采用欧式距离  
 index = np.argsort(length) #获取排序后的索引  
 top\_lable = [Train\_label[i] for i in index[:k]] #获取前k个最近邻点的标签  
 predict\_label = Counter(top\_lable).most\_common(1)[0][0] #投票，找出出现最多的点的标签  
 Sort\_label = np.append(Sort\_label,predict\_label)  
 #print(Sort\_label)  
 Pd\_A = np.array([[0,0]]) #分类后的A点集合  
 Pd\_B = np.array([[0,0]]) #分类后的B点集合  
 for i in range(500):  
 if Sort\_label[i] == **'A'**:  
 Pd\_A = np.append(Pd\_A,[TestP\_set[i]],axis=0)  
 elif Sort\_label[i] == **'B'**:  
 Pd\_B = np.append(Pd\_B,[TestP\_set[i]],axis=0)  
 #print(Pd\_A)  
 #print(Pd\_B)  
 #print(Sort\_label)  
 x1,y1 = Pd\_A.T  
 x2,y2 = Pd\_B.T  
 plt.axis()  
 plt.title(**"Predicted(k="**+str(k)+**')'**)  
 plt.xlabel(**"x"**)  
 plt.ylabel(**"y"**)  
 plt.scatter(x1,y1,label = **'A'**)  
 plt.scatter(x2,y2,c = **'r'**,label = **'B'**)  
 plt.legend()  
 plt.show()  
 #计算分类准确率  
 error = 0  
 for i in range(500):  
 if Sort\_label[i] != **'A'** and i<300:  
 error += 1  
 elif Sort\_label[i] != **'B'** and i>300:  
 error += 1  
 correct\_rate = 1 - (error/500)  
 print()  
 print(**'k='**+str(k)+**'时'**,**'分类正确率为：'**,correct\_rate)  
KNN(20,TrainP\_set,TestP\_set)  
KNN(15,TrainP\_set,TestP\_set)  
KNN(10,TrainP\_set,TestP\_set)  
KNN(5,TrainP\_set,TestP\_set)  
KNN(1,TrainP\_set,TestP\_set)