1. （必填）自己提出的问题的理解（罗列全部）：

问题一：knn算法K的取值是怎么确定的？

自己的理解：常用的方法是从k=1开始，使用检验集估计分类器的误差率。重复该过程，每次K增值1，允许增加一个近邻。选取产生最小误差率的K。一般k的取值不超过20，上限是n的开方，随着数据集的增大，K的值也要增大。

问题二：bagging什么时候会降低稳定分类器的准确率？

自己的理解：在稳定分类器本身效果不好，且相互之间差异较大，即“差而不同”的时候也。此时分类器的准确率可能会比最差的分类器还差。

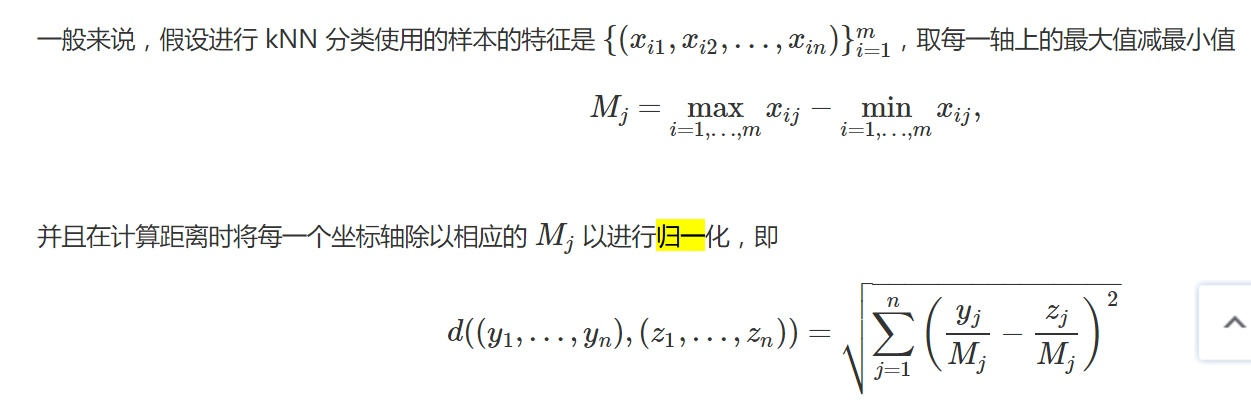
1. （必填）别人提出的问题的理解（选择几个问题罗列，并给出理解）：

问题三： k近邻学习有什么缺点？

样本不平衡的时候，对稀有类别的预测准确率低。当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的K个邻居中大容量类的样本占多数。

问题四：样本不同属性数值大小差距很大，那么我们该如何处理？

先归一化，使得每个特征同等重要，再进行分类。归一化的方法如下：



1. （必填）读书计划

1、本周完成的内容章节：第三章

2、下周计划：第四章

四、（选做）读书摘要及理解或伪代码的具体实现（读书摘要、伪代码的具体实现代码等可以写到这个部分）

1、读书摘要及理解（选做）

3.9 k-近邻算法

from numpy import \*

import operator as op

# 原始训练集

def createDataSet():

group = array([[1.0, 1.1], [1.0, 1.0], [0, 0], [0, 0.1]])

labels = ['A', 'A', 'B', 'B']

return group, labels

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

dataSetSize = dataSet.shape[0]

diffMat = tile(intX, (dataSetSize,1)) - dataSet

sqdiffMat = diffMat\*\*2

sqDistances = sqdiffMat.sum(axis=1)

distances = sqDistances\*\*0.5

sortedDistIndicies = distances.argsort()

classCount = {}

for i in range(k):

voteIlabel = labels[sortedDistIndicies[i]]

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

sortedClassCount = sorted(classCount.items(), key=op.itemgetter(1), reverse=True)

return sortedClassCount[0][0]