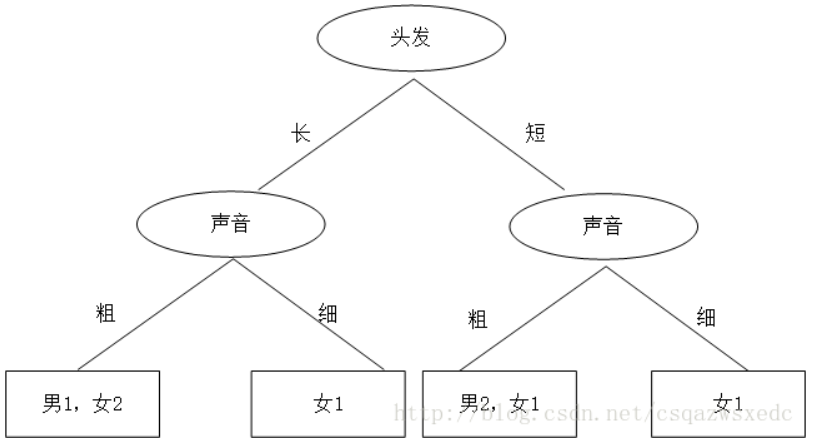
决策树

一天，老师问了个问题，只根据头发和声音怎么判断一位同学的性别。

为了解决这个问题，同学们马上简单的统计了7位同学的相关特征，数据如下：



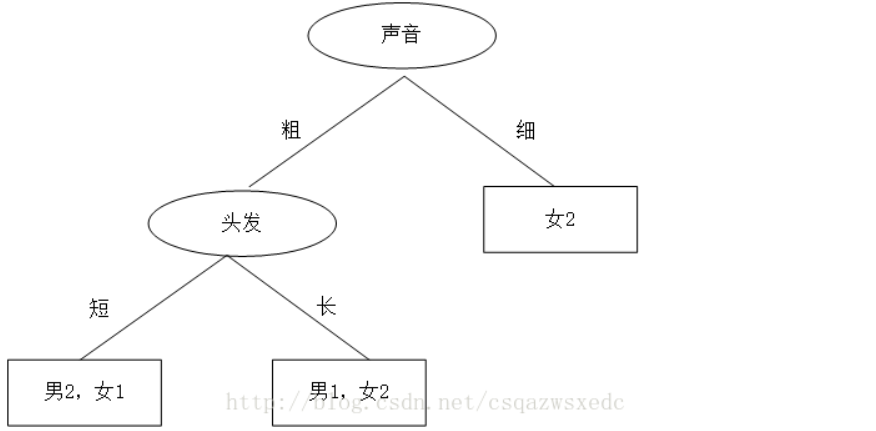
机智的贺嘉刘同学想了想，先根据头发判断，若判断不出，再根据声音判断，于是画了一幅图，如下：



于是，一个简单、直观的决策树就这么出来了。头发长、声音粗就是男生；头发长、声音细就是女生；头发短、声音粗是男生；头发短、声音细是女生。

原来机器学习中决策树就这玩意，这也太简单了吧。。。

这时又蹦出个同学刘嘉贺，想先根据声音判断，然后再根据头发来判断，如是大手一挥也画了个决策树：



同学刘嘉贺的决策树：首先判断声音，声音细，就是女生；声音粗、头发长是男生；声音粗、头发长是女生。

那么问题来了：同学贺嘉刘和同学刘嘉贺谁的决策树好些？计算机做决策树的时候，面对多个特征，该如何选哪个特征为最佳的划分特征？

划分数据集的大原则是：将无序的数据变得更加有序。

我们可以使用多种方法划分数据集，但是每种方法都有各自的优缺点。于是我们这么想，如果我们能测量数据的复杂度，对比按不同特征分类后的数据复杂度，若按某一特征分类后复杂度减少的更多，那么这个特征即为最佳分类特征。

Claude Shannon 定义了熵（entropy）和信息增益(information gain)。

用熵来表示信息的复杂度，熵越大，则信息越复杂。公式如下：

信息增益(information gain)，表示两个信息熵的差值。

首先计算未分类前的熵，总共有8位同学，男生3位，女生5位。

熵（总）=-3/8\*log2(3/8)-5/8\*log2(5/8)=0.9544

接着分别计算贺嘉刘和刘嘉贺分类后信息熵。

贺嘉刘首先按头发分类，分类后的结果为：长头发中有1男3女。短头发中有2男2女。

熵（贺嘉刘长发）=-1/4\*log2(1/4)-3/4\*log2(3/4)=0.8113

熵（贺嘉刘短发）=-2/4\*log2(2/4)-2/4\*log2(2/4)=1

熵（贺嘉刘）=4/8\*0.8113+4/8\*1=0.9057

信息增益（贺嘉刘）=熵（总）-熵（贺嘉刘）=0.9544-0.9057=0.0487

同理，按照刘嘉贺的方法，首先按声音特征来分，分类后的结果为：声音粗中有3男3女。声音细中有0男2女。

熵（刘嘉贺声音粗）=-3/6\*log2(3/6)-3/6\*log2(3/6)=1

熵（刘嘉贺粗）=-2/2\*log2(2/2)=0

熵（刘嘉贺）=6/8\*1+2/8\*0=0.75

信息增益（刘嘉贺）=熵（总）-熵（贺嘉刘）=0.9544-0.75=0.2087

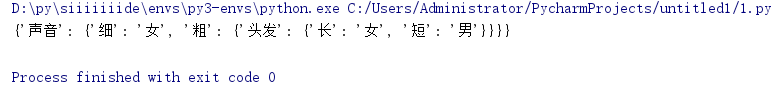
按刘嘉贺的方法，先按声音特征分类，信息增益更大，区分样本的能力更强，更具有代表性。

以上就是决策树ID3算法的核心思想。

接下来用python代码来实现ID3算法：

**from** math **import** log  
**import** operator  
  
**def** calcShannonEnt(dataSet): *# 计算数据的熵(entropy)* numEntries=len(dataSet) *# 数据条数* labelCounts={}  
 **for** featVec **in** dataSet:  
 currentLabel=featVec[-1] *# 每行数据的最后一个字（类别）* **if** currentLabel **not in** labelCounts.keys():  
 labelCounts[currentLabel]=0  
 labelCounts[currentLabel]+=1 *# 统计有多少个类以及每个类的数量* shannonEnt=0  
 **for** key **in** labelCounts:  
 prob=float(labelCounts[key])/numEntries *# 计算单个类的熵值* shannonEnt-=prob\*log(prob,2) *# 累加每个类的熵值* **return** shannonEnt  
  
**def** createDataSet1(): *# 创造示例数据* dataSet = [[**'长'**, **'粗'**, **'男'**],  
 [**'短'**, **'粗'**, **'男'**],  
 [**'短'**, **'粗'**, **'男'**],  
 [**'长'**, **'细'**, **'女'**],  
 [**'短'**, **'细'**, **'女'**],  
 [**'短'**, **'粗'**, **'女'**],  
 [**'长'**, **'粗'**, **'女'**],  
 [**'长'**, **'粗'**, **'女'**]]  
 labels = [**'头发'**,**'声音'**] *#两个特征* **return** dataSet,labels  
  
**def** splitDataSet(dataSet,axis,value): *# 按某个特征分类后的数据* retDataSet=[]  
 **for** featVec **in** dataSet:  
 **if** featVec[axis]==value:  
 reducedFeatVec =featVec[:axis]  
 reducedFeatVec.extend(featVec[axis+1:])  
 retDataSet.append(reducedFeatVec)  
 **return** retDataSet  
  
**def** chooseBestFeatureToSplit(dataSet): *# 选择最优的分类特征* numFeatures = len(dataSet[0])-1  
 baseEntropy = calcShannonEnt(dataSet) *# 原始的熵* bestInfoGain = 0  
 bestFeature = -1  
 **for** i **in** range(numFeatures):  
 featList = [example[i] **for** example **in** dataSet]  
 uniqueVals = set(featList)  
 newEntropy = 0  
 **for** value **in** uniqueVals:  
 subDataSet = splitDataSet(dataSet,i,value)  
 prob =len(subDataSet)/float(len(dataSet))  
 newEntropy +=prob\*calcShannonEnt(subDataSet) *# 按特征分类后的熵* infoGain = baseEntropy - newEntropy *# 原始熵与按特征分类后的熵的差值* **if** (infoGain>bestInfoGain): *# 若按某特征划分后，熵值减少的最大，则次特征为最优分类特征* bestInfoGain=infoGain  
 bestFeature = i  
 **return** bestFeature  
  
**def** majorityCnt(classList): *#按分类后类别数量排序，比如：最后分类为2男1女，则判定为男；* classCount={}  
 **for** vote **in** classList:  
 **if** vote **not in** classCount.keys():  
 classCount[vote]=0  
 classCount[vote]+=1  
 sortedClassCount = sorted(classCount.items(),key=operator.itemgetter(1),reverse=**True**)  
 **return** sortedClassCount[0][0]  
  
**def** createTree(dataSet,labels):  
 classList=[example[-1] **for** example **in** dataSet] *# 类别：男或女* **if** classList.count(classList[0])==len(classList):  
 **return** classList[0]  
 **if** len(dataSet[0])==1:  
 **return** majorityCnt(classList)  
 bestFeat=chooseBestFeatureToSplit(dataSet) *#选择最优特征* bestFeatLabel=labels[bestFeat]  
 myTree={bestFeatLabel:{}} *#分类结果以字典形式保存* **del**(labels[bestFeat])  
 featValues=[example[bestFeat] **for** example **in** dataSet]  
 uniqueVals=set(featValues)  
 **for** value **in** uniqueVals:  
 subLabels=labels[:]  
 myTree[bestFeatLabel][value]=createTree(splitDataSet\  
 (dataSet,bestFeat,value),subLabels)  
 **return** myTree  
  
  
**if** \_\_name\_\_==**'\_\_main\_\_'**:  
 dataSet, labels=createDataSet1() *# 创造示列数据* print(createTree(dataSet, labels)) *# 输出决策树模型结果*

输出结果为：



这个结果的意思是：首先按声音分类，声音细为女生；然后再按头发分类：声音粗，头发短为男生；声音粗，头发长为女生。

这个结果也正是刘嘉贺的结果。