from math import log

import operator

def entropy(data):

entries = len(data)

labels = {}

for feat in data:

label = feat[-1]

if label not in labels.keys():

labels[label] = 0

labels[label] += 1

entropy = 0.0

for key in labels:

probability = float(labels[key])/entries

entropy -= probability \* log(probability,2)

return entropy

def split(data, axis, val):

newData = []

for feat in data:

if feat[axis] == val:

reducedFeat = feat[:axis]

reducedFeat.extend(feat[axis+1:])

newData.append(reducedFeat)

return newData

def choose(data):

features = len(data[0]) - 1

baseEntropy = entropy(data)

bestInfoGain = 0.0;

bestFeat = -1

for i in range(features):

featList = [ex[i] for ex in data]

uniqueVals = set(featList)

newEntropy = 0.0

for value in uniqueVals:

newData = split(data, i, value)

probability = len(newData)/float(len(data))

newEntropy += probability \* entropy(newData)

infoGain = baseEntropy - newEntropy

if (infoGain > bestInfoGain):

bestInfoGain = infoGain

bestFeat = i

return bestFeat

def majority(classList):

classCount={}

for vote in classList:

if vote not in classCount.keys(): classCount[vote] = 0

classCount[vote] += 1

sortedClassCount = sorted(classCount.iteritems(), key=operator.itemgetter(1), reverse=True)

return sortedClassCount[0][0]

def tree(data,labels):

classList = [ex[-1] for ex in data]

if classList.count(classList[0]) == len(classList):

return classList[0]

if len(data[0]) == 1:

return majority(classList)

bestFeat = choose(data)

bestFeatLabel = labels[bestFeat]

theTree = {bestFeatLabel:{}}

del(labels[bestFeat])

featValues = [ex[bestFeat] for ex in data]

uniqueVals = set(featValues)

for value in uniqueVals:

subLabels = labels[:]

theTree[bestFeatLabel][value] = tree(split/(data, bestFeat, value),subLabels)

return theTree