# 基础实验指导与报告

姓名： 学号：

# 题目一：决策树

## 一、实验目的

掌握机器学习中决策树的基本原理，以及决策树的ID3算法实现。

## 实验内容

基于给定西瓜数据集实现决策树ID3算法。

## 三、实验要求

1. 创建数据集和ID3算法框架代码已给出，请**完成代码中的TODO部分**。
2. 说明解题思路，必要时给出公式推导。
3. 对实验结果进行总结分析，并绘制出所构建的决策树。
4. 提交实验时，上传代码源文件和实验报告。

## 四、实验步骤

### 1. 解题思路

### 2. 完成代码

import operator

from math import log

**# 创建数据集**

def createDataSet():

dataSet = [['青绿', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', '是'],

['乌黑', '蜷缩', '沉闷', '清晰', '凹陷', '硬滑', '是'],

['乌黑', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', '是'],

['青绿', '蜷缩', '沉闷', '清晰', '凹陷', '硬滑', '是'],

['浅白', '蜷缩', '浊响', '清晰', '凹陷', '硬滑', '是'],

['青绿', '稍蜷', '浊响', '清晰', '稍凹', '软粘', '是'],

['乌黑', '稍蜷', '浊响', '稍糊', '稍凹', '软粘', '是'],

['乌黑', '稍蜷', '浊响', '清晰', '稍凹', '硬滑', '是'],

['乌黑', '稍蜷', '沉闷', '稍糊', '稍凹', '硬滑', '否'],

['青绿', '硬挺', '清脆', '清晰', '平坦', '软粘', '否'],

['浅白', '硬挺', '清脆', '模糊', '平坦', '硬滑', '否'],

['浅白', '蜷缩', '浊响', '模糊', '平坦', '软粘', '否'],

['青绿', '稍蜷', '浊响', '稍糊', '凹陷', '硬滑', '否'],

['浅白', '稍蜷', '沉闷', '稍糊', '凹陷', '硬滑', '否'],

['乌黑', '稍蜷', '浊响', '清晰', '稍凹', '软粘', '否'],

['浅白', '蜷缩', '浊响', '模糊', '平坦', '硬滑', '否'],

['青绿', '蜷缩', '沉闷', '稍糊', '稍凹', '硬滑', '否']]

# 特征名称列表

features = ['色泽', '根蒂', '敲声', '纹理', '脐部', '触感']

return dataSet, features

**#TODO 计算数据集的信息熵(entropy)**

def calcShannonEnt(dataSet):

pass

**# 按某个特征分类后的数据**

def splitDataSet(dataSet, axis, value):

retDataSet = []

for featVec in dataSet:

if featVec[axis] == value:

# 在数据集中去掉已经分类的特征属性

reducedFeatVec = featVec[:axis]

reducedFeatVec.extend(featVec[axis+1:])

retDataSet.append(reducedFeatVec)

return retDataSet

**#TODO 选择最优的分类特征**

def chooseBestFeatureToSplit(dataSet):

pass

# 选取分类后样本数量最多的类别

def majorityCnt(classList):

classCount = {}

for vote in classList:

#python字典的key必须是不可变的比如说数字、元组、字符串，也因此列表不能当作key

if vote not in classCount.keys():

classCount[vote] = 0

classCount[vote] += 1

#reverse=true为降序排序

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

return sortedClassCount[0][0]

# 构建决策树(ID3算法)

def createTree(dataSet, features):

# 类别列表

classList = [example[-1] for example in dataSet]

# 若训练集中样本属于同一类别，则返回该类别

#count计算列表中某个相同元素的个数

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

return classList[0]

# 若特征属性已分类完毕，返回样本中出现次数最多的类别

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

return majorityCnt(classList)

# 选择最优特征的索引和标签

bestFeat = chooseBestFeatureToSplit(dataSet)

bestFeatLabel = features[bestFeat]

# 分类结果以字典形式保存

myTree = {bestFeatLabel:{}}

# 分类后训练集中分类特征会被删除，特征名称列表中对应特征名也应删除

#del 删除列表中的某个元素

del(features[bestFeat])

featValues = [example[bestFeat] for example in dataSet]

uniqueVals = set(featValues)

for value in uniqueVals:

subFeatures = features[:]

myTree[bestFeatLabel][value] = createTree(splitDataSet(dataSet, bestFeat, value), subFeatures)

return myTree

if \_\_name\_\_=='\_\_main\_\_':

# 创建数据集

dataSet, features = createDataSet()

# 输出决策树模型结果

print(createTree(dataSet, features))

### 3. 实验结果（绘制出构建的决策树）

### 4. 总结分析