

**Department of Computer Science and Engineering**

**(UG Studies)**

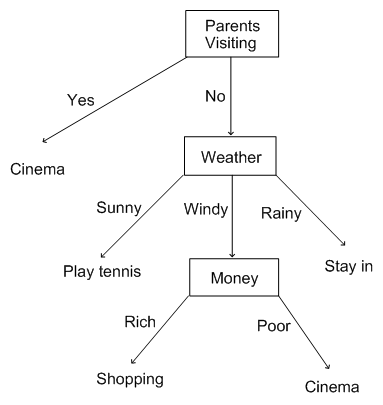
**PES University, Bangalore-560085**

|  |  |
| --- | --- |
| **Session :** Aug - Dec 2017  **Credits :** 0-0-2-0-1 | UE14CS405 : Machine Learning Lab |
| **Lab # :** 03 | Construct/Visualize a Decision Tree using Information Gain (Entropy) attributes on a (at least) 32- size data set with (at least) 8- Attributes |

**Theory:**

Imagine you only ever do four things at the weekend: go shopping, watch a movie, play tennis or just stay in. What you do depends on three things: the weather (windy, rainy or sunny); how much money you have (rich or poor) and whether your parents are visiting. You say to your yourself: if my parents are visiting, we'll go to the cinema. If they're not visiting and it's sunny, then I'll play tennis, but if it's windy, and I'm rich, then I'll go shopping. If they're not visiting, it's windy and I'm poor, then I will go to the cinema. If they're not visiting and it's rainy, then I'll stay in.

To remember all this, you draw a flowchart which will enable you to read off your decision. We call such diagrams decision trees. A suitable decision tree for the weekend decision choices would be as follows:



We can see why such diagrams are called trees, because, while they are admittedly upside down, they start from a root and have branches leading to leaves (the tips of the graph at the bottom). Note that the leaves are always decisions, and a particular decision might be at the end of multiple branches (for example, we could choose to go to the cinema for two different reasons). In ID3 Algorithm we use Entropy to generate decision trees.

**Dataset**

It contains five animals pulled from the sea and asks if they can survive without coming to the surface and if they have flippers. Classify these animals into two classes: fish and not fish. Decide whether we should split the data based on the first feature or the second feature. To answer this question, we need some quantitative way of determining how to split the data. Information gain or Entropy is used to find the attribute with highest Information gian, this attribute is taken as root node.

|  |  |  |
| --- | --- | --- |
| **can survive without coming to the surface** | **flippers** | **fish and not fish** |
| yes | yes | yes |
| yes | yes | yes |
| yes | no | no |
| no | yes | No |
| no | yes | no |

**CODE:**

from math import log

def majorityCnt(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]

#for calculting entropy

def calcShannonEnt(dataSet):

numEntries = len(dataSet)

labelCounts = {}

for featVec in dataSet:

currentLabel = featVec[-1]

if currentLabel not in labelCounts.keys():

labelCounts[currentLabel] = 0

labelCounts[currentLabel] += 1

shannonEnt = 0.0

#Complete the missing code to calculate entropy.

def createDataSet():

dataSet = [[1, 1, 'yes'],

[1, 1, 'yes'],

[1, 0, 'no'],

[0, 1, 'no'],

[0, 1, 'no']]

labels = ['no surfacing','flippers']

return dataSet, labels

myDat,labels=createDataSet()

def splitDataSet(dataSet, axis, value):

retDataSet = []

for featVec in dataSet:

if featVec[axis] == value:

reducedFeatVec = featVec[:axis]

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

retDataSet.append(reducedFeatVec)

return retDataSet

#choosing the best feature to split

def chooseBestFeatureToSplit(dataSet):

numFeatures = len(dataSet[0]) - 1

baseEntropy = calcShannonEnt(dataSet)

bestInfoGain = 0.0; bestFeature = -1

for i in range(numFeatures):

featList = [example[i] for example in dataSet]

uniqueVals = set(featList)

newEntropy = 0.0

for value in uniqueVals:

subDataSet = splitDataSet(dataSet, i, value)

prob = len(subDataSet)/float(len(dataSet))

newEntropy += prob \* calcShannonEnt(subDataSet)

infoGain = baseEntropy - newEntropy

if (infoGain > bestInfoGain):

bestInfoGain = infoGain

bestFeature = i

return bestFeature

print("the best feature to split is",chooseBestFeatureToSplit(myDat))

#function to build tree recursively

def createTree(dataSet,labels):

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

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

return classList[0]

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

return majorityCnt(classList)

bestFeat = chooseBestFeatureToSplit(dataSet)

bestFeatLabel = labels[bestFeat]

myTree = {bestFeatLabel:{}}

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

print(createTree(myDat,labels))

**TO DO:**

**1)** Understand Entropy , Identify the missing code and execute the code.

2) Draw the decision tree for the output manually.

3)Understand the code attached (no\_of\_rings\_predict.py )for 8 attributes**(Only Understanding)**

**Learning Outcomes:**

1. Decision tree builds a model called tree using training data, once the model is ready new test tuple with out the class label can be passed to the model and trace the path to get the label for the test data.

2. Decision Rules can be genrated from decision tree.

3. It is a supervised learning method for predicting class label