David Luong
Machine Learning CPSC 483
Professor Shilpa Lakhanpal
Nov 27, 2019

1)

```
import pandas as pd
import math
import numpy as np

class Node:
   def __init__(self, data):
       self.children = []
       self.data = data

   def getNode(self, i):
       return self.children[i]

   def insert(self, data):
       self.children.append(Node(data))

   def insertNode(self, node):
       self.children.append(node)

   def PrintTree(self):
       print(self.data)
       if self.children:
           print ("Children")
       for i in self.children:
           i.PrintTree()

def entrophy(x, y):
   ratio = x/y
   if ratio == 1.0 or ratio == 0.0:
       return 0
   else:
       return -(ratio * math.log2(ratio) + (1 - ratio) * math.log2(1-ratio))

def entrophyS(data):
```

```
    count = 0
    size = len(data)
    end = len(data.columns)-1
    label1 = data.iloc[0, end]

    for x in range(size):
        if (data.iloc[x, end] == label1):
            count += 1
    return entrophy(count, size)

def mostCommon(data, col):
    values = []
    count = []
    size = len(data)
    for x in range(size):
        i = -1
        for j in range(len(values)):
            if data[col][x] == values[j]:
                i = j
                break
        if i == -1:
            values.append(data[col][x])
            count.append(1)
        else:
            count[i] += 1
    return values[count.index(max(count))]

def IG(data, col, result):
    size = len(data)
    resultLabel = data[result][0] #assumes binary values

    values = []
    count = [] # number of postives rows
    labelSize = []
    for x in range(size):
        i = -1
        for j in range(len(values)):
            if data[col][x] == values[j]:
                i = j
```

```
                break
        if i == -1:
            values.append(data[col][x])
            labelSize.append(1)
            if data[result][x] == resultLabel:
                count.append(1)
            else:
                count.append(0)
        else:
            labelSize[i] += 1
            if data[result][x] == resultLabel:
                count[i] += 1

    #print(col)
    ig = 0
    for j in range(len(values)):
        #print(values[j], count[j], labelSize[j])
        ig += (labelSize[j] / size) * entrophy(count[j],labelSize[j])
    return [entrophyS(data) - ig, values]

def newData(data, col, att): #enter col to remove and att to select
    df = data.copy()
    df = df.loc[df[col] == att]
    #print(df)
    df = df.drop(col, axis = 1)
    #print(df)
    return df

def ID3(data, colNames, result):
    end = len(data.columns)-1
    print(data)
    if entrophyS(data) == 0:
        root = Node(data.iloc[0, end])
    elif len(colNames) == 1:
        root = Node(mostCommon(data, result))
    else:
        IG_vals = [0] * (len(colNames) - 1)
        values = []
```

```
        for i in range(0, len(colNames) - 1):
            ig = IG(data, colNames[i], result)
            IG_vals[i] = ig[0]
            values.append(ig[1])
        print(IG_vals, values)
        max_ = IG_vals.index(max(IG_vals))
        print ("Most IG: ",colNames[max_])

        root = Node (colNames[max_])

        for i in range(len(values[max_])):
            df = newData(data, colNames[max_], values[max_][i])
            print (df)
            if df.empty:
                name = mostCommon(data, colNames[max_])
                root.insert(name)
            else:
                newCol = colNames.copy()
                newCol.pop(max_)
                root.insertNode(ID3(df, newCol, result))
    return root

# MAIN
data = pd.read_csv('data.csv',  header = 0)

colNames = list(data.columns.values)

root = ID3(data,colNames, colNames[len(colNames)- 1])

root.PrintTree()
```

2)

Decision Tree

	VOTES	
NO /		\ YES
force-into		leave-alone

3)

HAS a JOB	HAS an INSURANCE	VOTES	ACTION
no	yes	yes	leave-alone
yes	no	yes	leave-alone
no	yes	no	force-into