NTNU
TDT4171 Artificial Intelligence Methods

Assignment 4 - Decision Tree Learning Algorithm

Mathilde Hotvedt

March 18, 2017

Contents

1 Introduction 1

2 Implementing the IMPORTANCE function 1

3 Decision tree learning 2

4 Discussion 9

1 Introduction

In this assignment we are supposed to implement the decision tree learning algorithm from the textbook in the course TDT4171, [1, p.702] by using two different versions of the IMPORTANCE function; one where the importance is allocated randomly and one that uses information gain. I will be using the programming language Python for my implementation.

2 Implementing the IMPORTANCE function

As we were to implement two versions of this function, I chose to write one function that switches which version to use by testing on the variable gainImportance. If the variable is false, I will give the importance, which attribute to split the tree on, randomly to any of the attributes. If this variable is True then we use gain importance calculation, which uses the reduction in Entropy from before and after splitting on one of the attributes to decide which attribute to split the tree on. The attribute which produces highest reduction in entropy is chosen. See the IMPORTANCE function in listing 1. Entropy is a measure of the uncertainty of a random variable, and if information about the random variable is achieved then the entropy will be reduced. Entropy is defined as

$$H(V) = -\sum_k P(v_k) log_2 P(v_k) B(q) \quad = -(q log_2 q + (1-q) log_2 (1-q)) \tag{2.1}$$

Where the last equation specify the formula for a binary variable which has values True or False as we do in this assignment. The random variable V has values v_k each with probability $P(v_k)$, and q is here the probability of the random variable being True. The IMPORTANCE function uses the information gain, which is the reduction in entropy to choose an attribute. This is specified as

$$Gain(A) = B(\frac{p}{p+n}) - \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B(\frac{p_k}{p_k + n_k}) \tag{2.2}$$

where d is the number of distinct values that attribute A can take, p_k is the number of examples with with classification 1 after splitting the examples on attribute A, and similarily is n_k is the number of examples with classification 2. p is the total number of examples with classification 1, and n is the total number of examples with classification 2.

```
def importance(attributes,examples,gainImportance):
  if not gainImportance:
    bestA = random.randint(0,len(attributes)-1);
    return bestA

  #Calculating the current entropy of the examples
  exOfClassValue1 = findExWithAttrValue(examples,-1,1)
  probClassValue1 = float(len(exOfClassValue1))/len(examples)
  currentEntropy = calcEntropy(probClassValue1)

  #Calculating the expected reduction in entropy for each remaining attribute
  #Calculate remainder for each subset E which attribute
  #A has divided eksamples into and sum them
  infoGain = []
  for A in attributes:
    remainder = 0.0
    for d in range(1,numOfValuesForAttr+1):
      E = findExWithAttrValue(examples,A,d)
      if E:
        p_k = len(findExWithAttrValue(E,-1,1))
        n_k = len(findExWithAttrValue(E,-1,2))
        probPositive = float(p_k)/len(E)
        #calculate entropy of these examples
```

```
        B_k = calcEntropy(probPositive)
        remainder += float(p_k+n_k)/len(examples)*B_k

    #information gain for attribute A
    infoGain.append(currentEntropy - remainder)
  bestA = infoGain.index(max(infoGain))
  return bestA
```

Listing 1: IMPORTANCE function

3 Decision tree learning

To learn my decision tree according to the algorithm in [1, p.702], I will need several additional functions such as calculating the PLURALITY-VALUE, checking if the examples all have the same class, a function finding examples with a certain attribute value and a way to construct my decision learning tree.
I chose to construct my decision tree with the help of a predefined class called treeNode, which will represent each internal node in the tree. It has a variable that keeps the information on which attribute this node has split the examples on, and then a dictionary of its "children", which are examples with the value of the split attribute (1 or 2) as a key, see listing 2 and an illustration in figure 3.1.

```
#Internal node in the tree, have split its children on the attribute splitOnAttr
#and have subnodes children
class treeNode:
  def __init__(self,attr):
    self.splitOnAttr = attr
    self.children = {}
```

Listing 2: A tree node

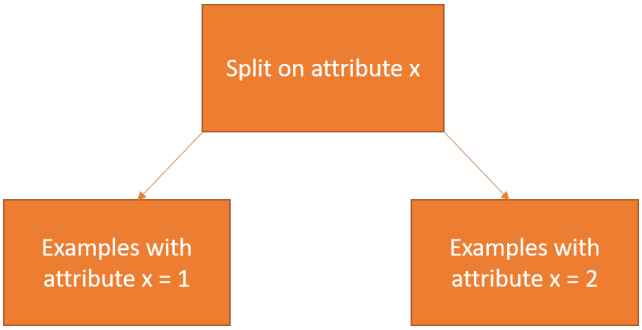

Figure 3.1: Illustration of a tree node

Further, the PLURALITY-VALUE function will be according to the definition in [1, p.700-701], returning the most common classification of the examples that where used to construct the parent node, see listing 3.The code for checking if all of the examples have same class is given in listing 4, the code for finding a subset of examples with a certain value for a certain attribute is given in listing 5 and finally, the learning algorithm is then given in listing 6.

```
#returning which class the majority of the examples are of
def pluralityValue(examples):
  classCount = [0]*numOfClass
  for ex in examples:
    classCount[ex[-1]-1] += 1
  return(classCount.index(max(classCount))+1)
```

Listing 3: PLURALITY-VALUE function

```
#Finds whether all of the examples have same classification or not
def sameClass(examples):
  classValue = examples[0][-1]
  for ex in examples:
    if ex[-1] is not classValue:
      return False
  return True
```

Listing 4: Check if examples are all of same class

```
#Finds a subset of the examples where attribute a have value v(1,2)
def findExWithAttrValue(examples,a,v):
  incidences = []
  for ex in examples:
    if ex[a] is v:
      incidences.append(ex)
  return incidences
```

Listing 5: Finding examples with attributes with certain value

```
#Creates a decision tree from the examples using algorithm specified in book.
#gainImportance = True calculates importance for attribute using entropy,
#False gives random allocation of importance
def decisionTreeLearning(examples,attributes,parent_eksamples,gainImportance):
  if not examples:
    return pluralityValue(parent_eksamples)
  if sameClass(examples):
    return examples[0][-1] #class spesification at back of vector
  if not attributes:
    return pluralityValue(examples)

  #Finds the most important attribute
  a = importance(attributes,examples,gainImportance)
  A = attributes[a]

  #Create a new decision tree
  tree = treeNode(A)
  #range over the different values for the attribute (1,2)
  #and create a new branch with a subtree
  for v in range(1,numOfClass+1):
    ex = findExWithAttrValue(examples,a,v)
    subTree = decisionTreeLearning(ex,attributes,examples,gainImportance)
    tree.children[v] = subTree
  return tree
```

Listing 6: Decision tree learning algorithm

The pseudocode for the algorithm given in [1, p.702] specifies that when you create the subtrees as in listing 6 line 23, you are supposed to send in a subset of the attributes, that is, remove the attribute A which the node has split its children on. However, when I do that, I end up with a tree that has only been able to have leaf nodes with classification 1 and has a lot less accuracy when testing it on the test examples. On the other hand, when using the same attribute list in every subtree I achieve trees with both classification 1 and 2 and a high accuracy. Therefore, I believe when I find sub examples in line 21 of listing 6 I have already made the attribute that the node is split on, less important, and my IMPORTANCE function using imformation gain will not choose this attribute anyways. The result of learning a tree using random importance is given in listing 7 and the result of using information gain as importance is given in listing 8.

```
Decision tree learning algorithm
Learning Random Tree...
```

```
Result of Tree:
Tree:
  Root: 5
  Root->1: 5
  Root->1->1: 3
  Root->1->1->1: 2
  Root->1->1->1->1: Classification 1
  Root->1->1->1->2: 4
  Root->1->1->1->2->1: 2
  Root->1->1->1->2->1->1: Classification 1
  Root->1->1->1->2->1->2: 6
  Root->1->1->1->2->1->2->1: 2
  Root->1->1->1->2->1->2->1->1: Classification 1
  Root->1->1->1->2->1->2->1->2: 6
  Root->1->1->1->2->1->2->1->2->1: 6
  Root->1->1->1->2->1->2->1->2->1->1: 2
  Root->1->1->1->2->1->2->1->2->1->1->1: Classification 1
  Root->1->1->1->2->1->2->1->2->1->1->2: 4
  Root->1->1->1->2->1->2->1->2->1->1->2->1: 6
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1: 5
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->1: 6
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->1->1: 5
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->1->1->1: 5
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->1->1->1->1: 6
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->1->1->1->1->1: 1
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->1->1->1->1->1->1: Classification 1
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->1->1->1->1->1->2: Classification 2
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->1->1->1->1->1->2: Classification 1
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->1->1->1->1->2: Classification 1
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->1->1->1->2: Classification 1
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->1->1->2: Classification 1
  Root->1->1->1->2->1->2->1->2->1->1->2->1->1->2: Classification 1
  Root->1->1->1->2->1->2->1->2->1->1->2->1->2: Classification 1
  Root->1->1->1->2->1->2->1->2->1->1->2->2: Classification 1
  Root->1->1->1->2->1->2->1->2->1->2->2: Classification 1
  Root->1->1->1->2->1->2->1->2->2: 2
  Root->1->1->1->2->1->2->2->1: Classification 2
  Root->1->1->1->2->1->2->2->2: 1
  Root->1->1->1->2->1->2->2->2->1: Classification 1
  Root->1->1->1->2->1->2->2->2->2: Classification 2
  Root->1->1->1->2->2: 7
  Root->1->1->1->2->2->1: 6
  Root->1->1->1->2->2->1->1: Classification 2
  Root->1->1->1->2->2->1->2: Classification 1
  Root->1->1->1->2->2->2: 6
  Root->1->1->1->2->2->2->1: Classification 2
  Root->1->1->1->2->2->2->2: Classification 1
  Root->1->1->2: 6
  Root->1->1->2->1: 2
  Root->1->1->2->1->1: 5
  Root->1->1->2->1->1->1: 2
  Root->1->1->2->1->1->1->1: 2
  Root->1->1->2->1->1->1->1->1: 6
  Root->1->1->2->1->1->1->1->1->1: 7
  Root->1->1->2->1->1->1->1->1->1->1: 2
  Root->1->1->2->1->1->1->1->1->1->1->1: 6
  Root->1->1->2->1->1->1->1->1->1->1->1->1: 6
  Root->1->1->2->1->1->1->1->1->1->1->1->1->1: 2
  Root->1->1->2->1->1->1->1->1->1->1->1->1->1->1: 1
  Root->1->1->2->1->1->1->1->1->1->1->1->1->1->1->1: Classification 1
  Root->1->1->2->1->1->1->1->1->1->1->1->1->1->1->2: Classification 2
```

```
Root->1->1->2->1->1->1->1->1->1->1->1->1->2: Classification 1
Root->1->1->2->1->1->1->1->1->1->1->1->1->2: Classification 1
Root->1->1->2->1->1->1->1->1->1->1->1->1->2: Classification 1
Root->1->1->2->1->1->1->1->1->1->1->1->2: Classification 1
Root->1->1->2->1->1->1->1->1->1->1->2: 5
Root->1->1->2->1->1->1->1->1->1->1->2->1: 7
Root->1->1->2->1->1->1->1->1->1->1->2->1->1: Classification 2
Root->1->1->2->1->1->1->1->1->1->1->2->1->2: 4
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1: 2
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1->1: 4
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1->1->1: 2
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1->1->1->1: 4
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1->1->1->1->1: 3
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1->1->1->1->1->1: Classification 1
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1->1->1->1->1->2: 1
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1->1->1->1->1->2->1: Classification 1
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1->1->1->1->1->2->2: Classification 2
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1->1->1->1->2: Classification 1
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1->1->1->2: Classification 1
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->1->1->2: Classification 1
Root->1->1->2->1->1->1->1->1->1->1->2->1->2->2: Classification 2
Root->1->1->2->1->1->1->1->1->1->1->2->2: Classification 2
Root->1->1->2->1->1->1->1->1->1->2: Classification 1
Root->1->1->2->1->1->1->1->1->2: Classification 1
Root->1->1->2->1->1->1->2: Classification 1
Root->1->1->2->1->2: Classification 1
Root->1->1->2->2: 2
Root->1->1->2->2->1: 1
Root->1->1->2->2->1->1: Classification 1
Root->1->1->2->2->1->2: Classification 2
Root->1->1->2->2->2: Classification 1
Root->1->2: Classification 1
Root->2: 3
Root->2->1: 7
Root->2->1->1: 7
Root->2->1->1->1: 1
Root->2->1->1->1->1: Classification 1
Root->2->1->1->1->2: 7
Root->2->1->1->1->2->1: 5
Root->2->1->1->1->2->1->1: Classification 2
Root->2->1->1->1->2->1->2: 3
Root->2->1->1->1->2->1->2->1: 1
Root->2->1->1->1->2->1->2->1->1: Classification 2
Root->2->1->1->1->2->1->2->1->2: 3
Root->2->1->1->1->2->1->2->1->2->1: 6
Root->2->1->1->1->2->1->2->1->2->1->1: 1
Root->2->1->1->1->2->1->2->1->2->1->1->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2: 4
Root->2->1->1->1->2->1->2->1->2->1->1->2->1: 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2: 6
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1: 6
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1: 7
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1: 4
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1: 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2: 5
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2: 5
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->1: Classification 1
```

```
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->2: 5
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->2->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->2->2: 5
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->2->2->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->2->2->2: 3
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->2->2->2->1: 3
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->2->2->2->1->1: 2
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->2->2->2->1->1->1:
  Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->2->2->2->1->1->2:
  Classification 2
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->2->2->2->1->2: Classification
  1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->1->2->2->2->2->2->2: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->1->2: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->1->2: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->1->2: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->1->2->2: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2: 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2: 3
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1: 3
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1: 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2: 7
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1: 5
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2: 4
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->2: 4
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->2->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->2->2: 5
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->2->2->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->2->2->2: 6
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->2->2->2->1: 4
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->2->2->2->1->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->2->2->2->1->2: 2
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->2->2->2->1->2->1: Classification
  1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->2->2->2->1->2->2: Classification
  2
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->1->2->2->2->2->2: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->1->2->2: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->1->2: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->1->2->2->2->2: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->2->1: 7
Root->2->1->1->1->2->1->2->1->2->1->2->1: 7
Root->2->1->1->1->2->1->2->1->2->1->2->1->1: 1
Root->2->1->1->1->2->1->2->1->2->1->2->1->1->1: Classification 2
Root->2->1->1->1->2->1->2->1->2->1->2->1->1->2: 5
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->1: Classification 2
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->2: 4
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->2->1: 6
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->2->1->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->2->1->2: 5
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->2->1->2->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->2->1->2->2: 7
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->2->1->2->2->1: 2
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->2->1->2->2->1->1: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->2->1->2->2->1->2: Classification 2
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->2->1->2->2->2: Classification 1
Root->2->1->1->1->2->1->2->1->2->1->2->1->2->2->1->2->2->2: Classification 2
```

```
Root->2->1->1->1->2->1->2->1->2->1->2->1->2: Classification 2
Root->2->1->1->1->2->1->2->1->2->1->2->2: Classification 2
Root->2->1->1->1->2->1->2->1->2->2: Classification 2
Root->2->1->1->1->2->1->2->2: Classification 2
Root->2->1->1->1->2->2: Classification 2
Root->2->1->1->2: Classification 1
Root->2->1->2: 6
Root->2->1->2->1: 1
Root->2->1->2->1->1: Classification 1
Root->2->1->2->1->2: 4
Root->2->1->2->1->2->1: 5
Root->2->1->2->1->2->1->1: Classification 1
Root->2->1->2->1->2->1->2: 4
Root->2->1->2->1->2->1->2->1: 3
Root->2->1->2->1->2->1->2->1->1: 1
Root->2->1->2->1->2->1->2->1->1->1: Classification 1
Root->2->1->2->1->2->1->2->1->1->2: 7
Root->2->1->2->1->2->1->2->1->1->2->1: Classification 1
Root->2->1->2->1->2->1->2->1->1->2->2: 1
Root->2->1->2->1->2->1->2->1->1->2->2->1: Classification 1
Root->2->1->2->1->2->1->2->1->1->2->2->2: 2
Root->2->1->2->1->2->1->2->1->1->2->2->2->1: Classification 1
Root->2->1->2->1->2->1->2->1->1->2->2->2->2: Classification 2
Root->2->1->2->1->2->1->2->1->2: Classification 1
Root->2->1->2->1->2->1->2->2: Classification 1
Root->2->1->2->1->2->2: 6
Root->2->1->2->1->2->2->1: 2
Root->2->1->2->1->2->2->1->1: Classification 1
Root->2->1->2->1->2->2->1->2: Classification 2
Root->2->1->2->1->2->2->2: Classification 1
Root->2->1->2->2: 2
Root->2->1->2->2->1: Classification 1
Root->2->1->2->2->2: 1
Root->2->1->2->2->2->1: Classification 1
Root->2->1->2->2->2->2: Classification 2
Root->2->2: 1
Root->2->2->1: Classification 1
Root->2->2->2: 6
Root->2->2->2->1: 6
Root->2->2->2->1->1: 5
Root->2->2->2->1->1->1: Classification 1
Root->2->2->2->1->1->2: 6
Root->2->2->2->1->1->2->1: 7
Root->2->2->2->1->1->2->1->1: 3
Root->2->2->2->1->1->2->1->1->1: Classification 1
Root->2->2->2->1->1->2->1->1->2: 1
Root->2->2->2->1->1->2->1->1->2->1: Classification 1
Root->2->2->2->1->1->2->1->1->2->2: 1
Root->2->2->2->1->1->2->1->1->2->2->1: Classification 1
Root->2->2->2->1->1->2->1->1->2->2->2: 7
Root->2->2->2->1->1->2->1->1->2->2->2->1: 3
Root->2->2->2->1->1->2->1->1->2->2->2->1->1: Classification 1
Root->2->2->2->1->1->2->1->1->2->2->2->1->2: 7
Root->2->2->2->1->1->2->1->1->2->2->2->1->2->1: 5
Root->2->2->2->1->1->2->1->1->2->2->2->1->2->1->1: Classification 1
Root->2->2->2->1->1->2->1->1->2->2->2->1->2->1->2: 2
Root->2->2->2->1->1->2->1->1->2->2->2->1->2->1->2->1: Classification 2
Root->2->2->2->1->1->2->1->1->2->2->2->1->2->1->2->2: Classification 1
Root->2->2->2->1->1->2->1->1->2->2->2->1->2->2: Classification 1
Root->2->2->2->1->1->2->1->1->2->2->2->2: Classification 1
Root->2->2->2->1->1->2->1->2: 1
Root->2->2->2->1->1->2->1->2->1: Classification 1
```

```
  Root->2->2->2->1->1->2->1->2->2: 7
  Root->2->2->2->1->1->2->1->2->2->1: Classification 1
  Root->2->2->2->1->1->2->1->2->2->2: 1
  Root->2->2->2->1->1->2->1->2->2->2->1: Classification 1
  Root->2->2->2->1->1->2->1->2->2->2->2: 2
  Root->2->2->2->1->1->2->1->2->2->2->2->1: Classification 2
  Root->2->2->2->1->1->2->1->2->2->2->2->2: Classification 1
  Root->2->2->2->1->1->2->2: Classification 1
  Root->2->2->2->2->1->2: Classification 1
  Root->2->2->2->2->2: 7
  Root->2->2->2->2->2->1: 2
  Root->2->2->2->2->2->1->1: Classification 2
  Root->2->2->2->2->2->1->2: Classification 1
  Root->2->2->2->2->2->2: Classification 2
Testing the tree we achieved correct classification on 20 examples out of 28:
Thereby giving an accuracy of 71.428571
```

Listing 7: Result of decision tree learning algorithm with random importance

```
Learning Information Gain Tree...
Result of Tree
Tree:
  Root: 1
  Root->1: Classification 1
  Root->2: 5
  Root->2->1: 3
  Root->2->1->1: 2
  Root->2->1->1->1: Classification 1
  Root->2->1->1->2: Classification 2
  Root->2->1->2: 2
  Root->2->1->2->1: Classification 2
  Root->2->1->2->2: Classification 1
  Root->2->2: 6
  Root->2->2->1: 2
  Root->2->2->1->1: 3
  Root->2->2->1->1->1: Classification 1
  Root->2->2->1->1->2: Classification 2
  Root->2->2->1->2: 3
  Root->2->2->1->2->1: Classification 2
  Root->2->2->1->2->2: Classification 1
  Root->2->2->2: 4
  Root->2->2->2->1: 2
  Root->2->2->2->1->1: 3
  Root->2->2->2->1->1->1: Classification 1
  Root->2->2->2->1->1->2: Classification 2
  Root->2->2->2->1->2: 3
  Root->2->2->2->1->2->1: Classification 2
  Root->2->2->2->1->2->2: Classification 1
  Root->2->2->2->2: 2
  Root->2->2->2->2->1: Classification 2
  Root->2->2->2->2->2: 3
  Root->2->2->2->2->2->1: Classification 2
  Root->2->2->2->2->2->2: Classification 1
Testing the tree we achieved correct classification on 26 examples out of 28:
Thereby giving an accuracy of 92.857143
```

Listing 8: Result of decision tree learning algorithm with information gain importance

4 Discussion

As can be seen from the two results in listing 7 using random importance and listing 8 using information gain importance, there is a big difference in the size of the tree obtained when learning it. The classification of the test examples and the code to test the tree may be viewed in listing 9.

```
def test(examples,tree):
  correctClassification = 0
  for ex in examples:
    correctClass = ex[-1]
    node = tree
    estimatedClass = 0
    while True:
      if isinstance(node,int):
      #the leaves in the tree are nodes made of only one int, no children
        estimatedClass = node
        break
      A = node.splitOnAttr
      valueAttr = ex[A]
      if valueAttr not in node.children.keys():
        #we have no children, remember children is a dictionary (valueAttr,subtree)
        break
      else:
        node = node.children[valueAttr] #continue traversing tree
    if estimatedClass == correctClass:
      correctClassification += 1
  probCorrect = float(correctClassification)/len(examples)
  print("Testing the tree we achieved correct classification on %i examples out of %i: " %(
    correctClassification,len(examples)))
  print("Thereby giving an accuracy of %f" %(probCorrect*100))
```

Listing 9: Test function function

The tree from using information gain importance is compact and small and achieves an accuracy of 92.9%, whereas the tree from random importance allocation is very large and only achieves an accuracy of 71.4% during this run. From this result only I conclude that using information gain importance allocation is better than random importance allocation.
However, for every new run of the program, the random tree changes and therefore also the accuracy. Acctually, at some point, I obtained an accuracy for the random tree almost equal to the accuracy of the information gain tree. This is expected as the random allocation does indeed randomly choose which attribute to split on each run. The information gain tree on the other hand, will stay exactly the same for each run due to having the same examples in each run and the information gain will be calculated the same way, resulting in the same result each time.
One reason for choosing information gain importance allocation and not random is that random allocation is often prune to overfitting the example data. This is due to the algorithm by nature, tries to fit the tree to all examples in the best possible way. That also makes it vulnerable to measurment noise and errors, that is, examples which has been classified wrongly because the data has been changed. Information gain importance allocation however, takes this into account when learning the tree and minimizes the remaining information needed to classify new examples, making it robust against measurement noise and errors. However, as also seen from the accuracy gained, the tree is only learned from a certain amount of examples and will therefore not give a perfect classification. More examples to train the tree on before testing it might have achieved a better accuracy, but then again, large amount of examples requires large amount of time to train the tree.
The code used in this example may be located at
`https://github.com/mathildh/TDT4171---Methods-in-artifical-intelligence`

REFERENCES

[1] Stuart Russel, Peter Norvig, *Artificial Intelligence: A Modern Approach*, 3rd edition, Prentice Hall, 2010