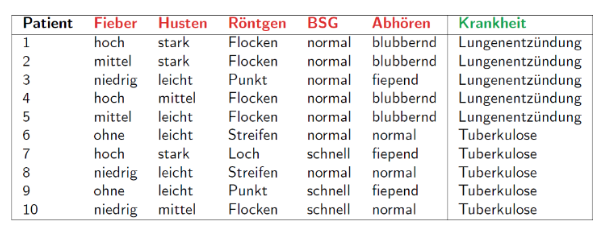
Algo analysis



Phase 1:

1. Current score (entropy) = 1 calc entropy of labels {Lung:5, TBC:5}

2. for column 0:

['hoch', 'mittel', 'niedrig', 'hoch', 'mittel', 'ohne', 'hoch', 'niedrig', 'ohne', 'niedrig']

For hoch (1st value in 1st column):

Divide to set 1( rows containing hoch) & set 2 (not containing hoch)

P = len(set1) / len(rows) = 3 /10 = 0.3 (occurrence of "hoch" within rows)

(max)Gain ("hoch") = currScore (1) – p(0.3) \*entropy(set1) – (1-p)(0.7)

entropy(set2) = 0.034

**Tree building**

**basically we look for the attribute with the biggest info gain = expected reduction in impurity (entropy). We store the max gain, the attribute col, and the split data set which serve as the sons (left = set1, right = set 2 – arbitrary). Next, the same process is done for the for the data set on the sons until we reach gain==0 (impurity==0) == leaf == no entropy == clear cut decision**

1. calculate curr data set impurity (gini/entropy) = c (entropy of label) (lung:5, tbc: 5) ->

-sig(p\*log2p) = -0.5\*log0.5-0.5log0.5 = 1 (entropy demonstrated)

2. for each value x (candidate attribute) in each column and each row (traversing like a snake on all columns) in the current dataset (current since it is replaced each recursion loop, each time new dataset from the divided one. Each dataset belongs to right or left branch) : divide dataset (rows) into 2 sets: set 1.where x exists, set 2. where it doesn't. when dealing with ints we check of the attribute is <= to the value at the same location(column) of incoming rows (greater values go to set 1, smaller go to set 2)

3. calculate p(set 1) = set1/all = p

4. gain (metric) = c – p\* set A entropy (child left) – (1-p) set B entropy (child right) (like in step 1) (p chance of choosing label 1, 1-p chance of choosing label 2)

If gain is the biggest – save attribute's value and col (used for classifying), save both sets (would serve as children),

The attribute with max gain will be the node. The node will store col,value, left,right branches, results(leaf node) explained below

5. now we know the best gain attribute + 2 of it's dataset sons! (but we don't know which attributes the sons represent – to find out, we recurse for left and right – the same process but with the different dataset (set 1 for left, set 2 for right).

If best gain == 0 – it means the entropy is 0, meaning we have a clear cut answer == meaning a leaf node has bean found. The node of the tree will store the result in tree.result (the only unique label in curr dataset, and when there is only one unique label in data set the entropy is 0). **This occurs when the dataset has gone so many splits, by choosing the best info gain, that only single label has stayed, causing 0 entryop.**

**Tree Classify**

When dealing with strings

1. input prediction (row==array)

2. if the curr node attribute came from the same column and it's equal in value and location to the incoming attribute in input – go to true branch (with strings its == , with values >=) the column matters, since it denotes the feature. for example normal may appear in col 1 and 3, thus we want to know if the attribute in the node refers to which column - important for decision making (tree traversal). The attribute in node and input are identical if they are the same and came from the same column (feature)!

3. Else go to false branch and repeat the process

Do the process until tree.result is not null – it contains the answer (lung / tbc) and it will be returned