**World Quant University**

**Professor: Douglas Kelly**

**Algorithms I**

Nikolas Lippmann Pareschi - [nikolaslippmann@gmail.com](mailto:nikolaslippmann@gmail.com)

**FINAL PROJECT: ALGORITHMS I**

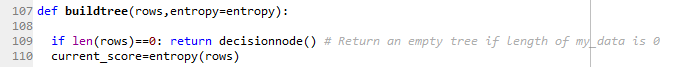
The code that I have used is based on our Final Project Lab and in the code provided by book “Programming Collective Intelligence”, written by Toby Segaran. We will present 2 trees. One intended in the console and one in a binary tree graph.

These instructions were provided in the piazza group. Below each instruction I will post the related code and explanations. After we give the interpretation of results.

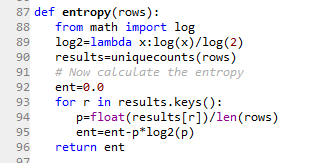
"""

Procedures for Building a Tree:

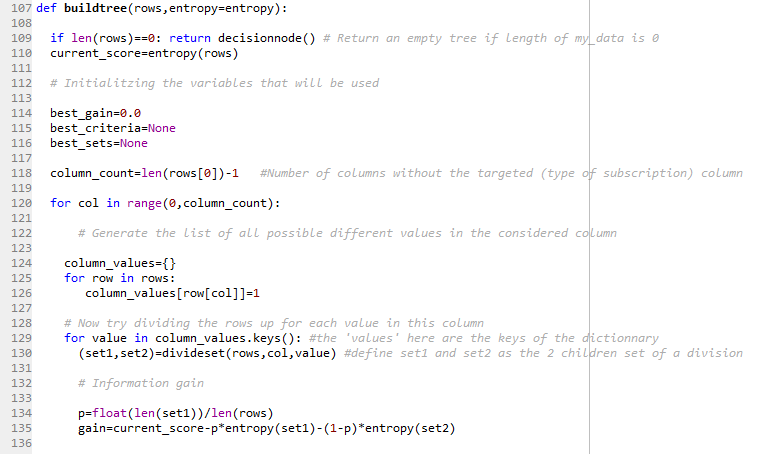
1. Create and return an empty tree if length of my\_data is 0



2. Calculate the info gain and save (use entropy(my\_data))



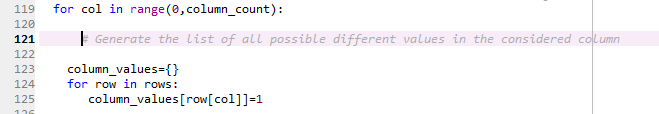
We call the entropy function in the function used to build the tree and then we save to the variable *gain*.



3. Identify the set of unique values in each column (except the for decision column). Fr example, the set of unique values for country would be {'UK', 'France', 'New Zealand', 'USA'} for referring website or source, it would be {'google', '(direct)', 'slashdot', 'digg', 'kiwitobes'} and for page views it would be {12, 18, 19, 21, 23, 24}. Lastly, there would also be one for yes/no column like so - {'yes', 'no'}

Do not do for the decision column which is Basic, Premium or Known because this is the decision criteria





We use this to create an entry in the dictionary. The values do not matter, only the keys. We could have used any number besides 1. We subtract 1 from column\_count to not compute the decision criteria (Basic, Premium or None).

4. Calculate the info gain for each division set:

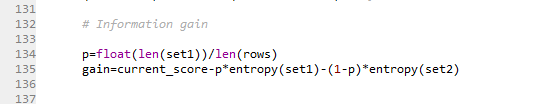
According to wikipedia:

Info Gain = Entropy(parent) - Weighted Sum of Entropy(Children)

or

prob = float(len(set1)/(len(set1)+len(set2)))

info\_gain = entropy(my\_data)-prob\*entropy(set1)-(1-prob)\*entropy(set2)

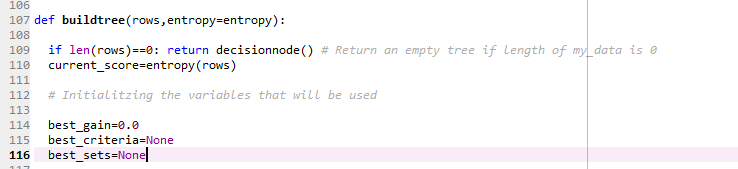


5. Recall the following variables in the starter code:

best\_gain=0.0

best\_criteria=None

best\_sets=None



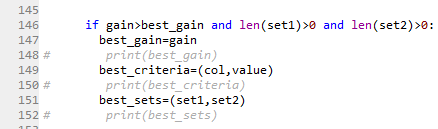
6. For each divide set if info gain is better than best\_gain, then

best\_gain=new information gain

best\_criteria=(col, value) that resulted in the best gain (attribute and value)

e.g (country, 'USA') or (source, 'google')

best\_sets=(set1,set2) #the split that resulted in the best gain

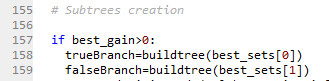


7. After best set known, if best\_gain > 0:

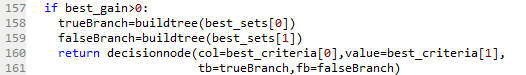
Create left and right substrees using recursion:

left = buildree(best\_sets[0])

right = buildree(best\_sets[1])



8. Return decision nodes using the best\_criteria and left/right tree

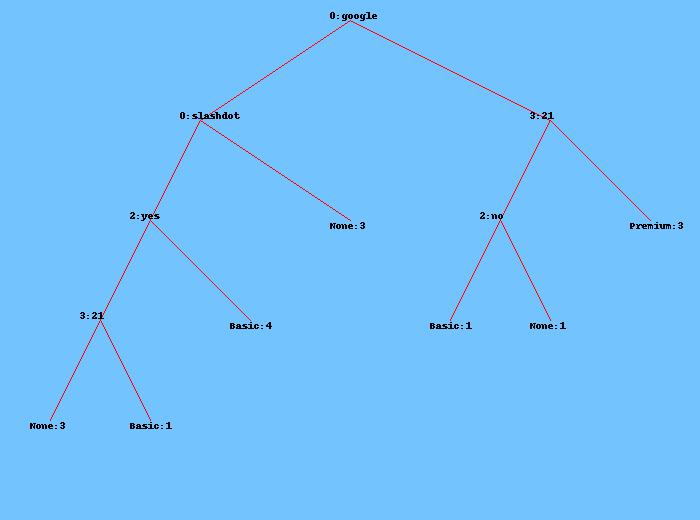


9. Otherwise return decision nodes built on uniquecounts(my\_data)



"""

We then used the code provided by World Quant University to build the tree:



Interpretation of the tree:

Considering that decision branches were build accordingly to maximize the information gain, i.e, selecting the best splits that minimized entropy, forming the most homogeneous group, we notice that the dimensions: source, number of views and FAQ (yes or no) are the most important. The countries do not bring significant information gain, which makes sense. All the countries are from the western civilization (New Zealand is geographically placed east but it shares the western culture), have a high PIB per capita and are well developed economies so we expect a high correlation among them. The FAQ is present in the 2 bigger branches and we can understand its importance. If the user has read all the FAQ his interest will be probably high. Number of views is another great metric present in 2 branches. The higher the number of views, the higher the interest. Google being the first decision criterium is also very interesting, maybe its importance comes from the fact that the users actively searched the theme.

We have also made a simpler tree, using indendation:

