HW06 : Decision Tree Implementation

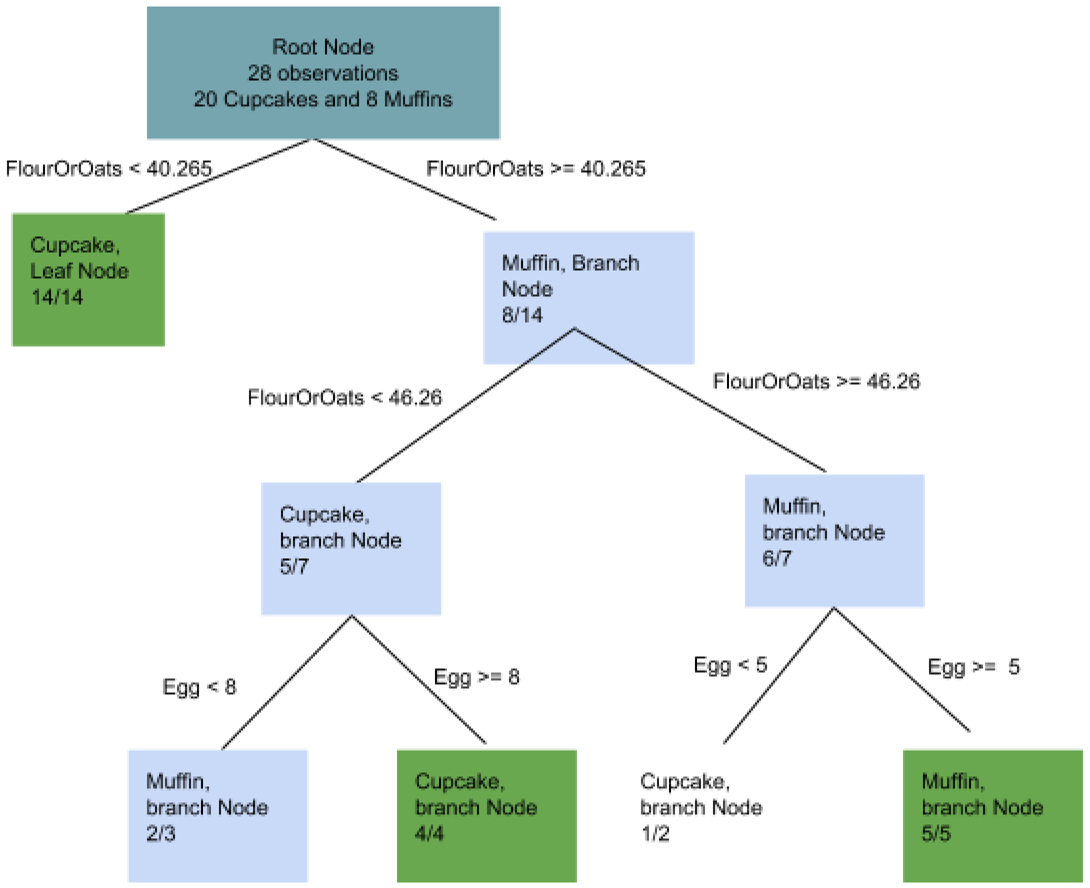
1.) What stopping criteria did you use?
I used two different conditions

1.) Number of nodes should be less than 8
2.) If the accuracy is 98% of a branch is 98% then stop the further split.

2.) Did you use any pruning or post-pruning?
No, I didn't do any pre or post pruning specially because I observed the information gain on each branch was significant therefore I didn't have to prune the tree.

3.) What splitting decision were you using?
I basically checked for median value for each attribute and calculated the weighted Gini Index to find the best attribute for split (In this case I tried to minimize the Weighted Gini Index)

4.) What structure did your final decision tree classifier have?
It is a binary tree where I have 8 Branches. Here I have three leaf nodes.



5.) What was the if-else tree you got? (Copy it into your write-up for completeness.)
I basically used the column indexes to avoid the spell mistakes or matching the actual column names in future for validations.

6.) In the following code I initially check for FlourOrOats (index for flour is [2]) content to be less than 40.285. If that is the true then the classification will be "Cupcake".

Suppose if the FlourOrOats is greater than 40.285 then I check whether FlourOrOats is less than 46.26. If the FlourOrOats is less then Number of egges is less than 8. If it is less than 8 then it is Muffin. If it is greater than 8 then it is Cupcake.

Suppose if the FlourOrOats is greater than 46.26 then check if the number of egg is less than 5. If it is less than 5 then the classification is cupcake. If it is greater than 5 then it is Muffin.

```
if(inputVal[2] < 40.285){
            print(c(id, ": classification is ", "cupcake", " -LN"))
      }else if(inputVal[2] < 46.26){
            if(inputVal[6]<8){
                  print(c(id, ": classification is ", "Muffin", " - RN-LN-LN"))
            }else {
                  print(c(id, ": classification is ", "cupcake", " -
RN-LN-RN"))
            }
      }else{
            if (inputVal[6] < 5){ # RN-RN
                  print(c(id, ": classification is ", "cupcake", " -
RN-RN-LN"))
            }else{
                  print(c(id, ": classification is ", "Muffin", " - RN-RN-RN"))
            }
      }
```

7.) Run the original training data back through your classifier. What was the accuracy of your resulting classifier, on the training data?

(26/28) *100 = 92.85%

8.) Did your program actually create the classifier program, or did it just generate the attribute list and thresholds for you to hand-code in.
Yes, My program created the classifier program. Please refer to
Line 195 for emitHeader <-function()
Line 220 for emitClassifier <- function()
Line 256 for emitFooter <- function()
Line 285 for emitDecisionTree <- function()
Finally Line 317 for emitDecisionTree() getting called inside the main function

9.) What else did you learn along the way here?

I learned a lot about R. In fact, I haven't done much recursive programming recently, this is a great refresher for that. Also I understand the how to use the global variables in R. I try my best to avoid this scenario. For an example I thought of writing the output into a file and read from the file, but I was thinking about reading time. It is always much faster to read from memory also this tree is wasn't that big therefore the memory usage wasn't really bad. I also learn the importance of data cleaning. Specially when I checked the initial data.. I was bit shocked with how many duplicates! I personally love cooking, therefore when I checked the recipes where vanila or baking powder is equal or greater than FlourOrOat contents, I was thinking there is an issue with the data collection.

10.) What can you conclude?
In my previous classes we used decision trees blindly without knowing the internals architecture. This is a great eye opening exercise. In fact I really enjoyed developing this decision tree and trying to find the places where I can optimize. Specially I would like to invest bit more time on data cleaning aspect. I think I spent a good amount of time on data cleaning. Also I am thinking about people create recipes with sugar free contents or use mostly brown sugar or Agave. How can I combine them to make up the classification more accurate!

Output for Validation Data:

```
Index              Classification                   Tree branch Qual
[1] "1"          ": classification is " "cupcake"     " - RN-LN-RN"
[1] "2"          ": classification is " "cupcake"     " -LN"
[1] "3"          ": classification is " "cupcake"     " - RN-RN-LN"
[1] "4"          ": classification is " "cupcake"     " -LN"
[1] "5"          ": classification is " "Muffin"    " - RN-RN-RN"
[1] "6"          ": classification is " "cupcake"     " -LN"
[1] "7"          ": classification is " "cupcake"     " - RN-RN-LN"
[1] "8"          ": classification is " "cupcake"     " -LN"
[1] "9"          ": classification is " "cupcake"     " - RN-RN-LN"
[1] "10"          ": classification is " "cupcake"     " -LN"
[1] "11"          ": classification is " "Muffin"    " - RN-RN-RN"
[1] "12"          ": classification is " "cupcake"     " -LN"
[1] "13"          ": classification is " "Muffin"    " - RN-RN-RN"
[1] "14"          ": classification is " "cupcake"     " -LN"
[1] "15"          ": classification is " "Muffin"    " - RN-RN-RN"
[1] "16"          ": classification is " "cupcake"     " -LN"
[1] "17"          ": classification is " "Muffin"    " - RN-RN-RN"
```

```
[1] "18"                   ": classification is " "cupcake"              " - RN-LN-RN"
[1] "19"                   ": classification is " "Muffin"              " - RN-RN-RN"
[1] "20"                   ": classification is " "cupcake"              " -LN"
```