Checkpoint 4
April 18 2017

In this checkpoint, you will be implementing your own decision tree. This decision tree is a simplified version of the one we called in Python scikit-learn. Our inputs and output interface should be the same as the one used in the scikit-learn. This will be implemented in the myTree.py file. You are free to make any additional methods in **MyDecisionTree** that are need to complete the below tasks.

Task 1 Metric.
The most simplified decision trees are based on either entropy or regression. The one here will be based on entropy. If you are interested in regression based tree, *The Elements of Statistical Learning* by Hastie, Tibshirani, and Friedman is a great reference. I have posted a reference for the entropy on Moodle. However, knowing the theory is not needed to implement the algorithm, and knowing the theory does not result in implementing a good algorithm. Turning theory into good implementations of algorithms is a skill that you as a computer scientist will have to slowly learn through hours of implementing algorithms over and over again! Luckily, I am going to try to walk you through this one as much as I can in words.

Entropy defines how much an attribute is pure/impure (unrelated/related) to other attributes. The goal is to find an attribute (e.g., date, number of words, etc) that defines the object the most and not bias from other attributes. Consider a case where there are two attributes {0,1}, and we want to find the entropy of the data relative to these two attributes. Assume there are 3 {0} elements and 3 {1} elements. We would define entropy as:
$Entropy(\{0,1\}) = -p_{\{0\}} lg\ p_{\{0\}} - p_{\{1\}} \lg p_{\{1\}} = -\frac{3}{6}\lg\frac{3}{6} - \frac{3}{6}\lg\frac{3}{6} = 1.$
Here we see, the entropy is 1 because we are not able to discern any information from the attribute that is evenly distributed. As the value of entropy get smaller, we are able to discern more.

We will first want to write a method that will calculate the entropy for a particular input row of a 2D Python list containing data (Given the 2DList and a number indicating the column). We will assume that the data in the column is numeric and discrete. That is, we will have as many attributes as unique values in the column. This method should return both a list of unique attributes and entropy for the column.

Task 2 Strip Row and Column.
You will make a method that will remove a given column from 2DList and split it into smaller 2DLists based on the unique elements in the column. See lecture notes for more details.

Task 3 Build Algorithm.
In the Python scikit-learn, they use the ID3 Algorithm. We are going to simplify the build algorithm a little bit to make it easier to implement. This algorithm will be called by the train interface. The algorithm is recursive as outline below. This algorithm is cannot be translated line to line, but give you a good insight into the steps required.

Alg Outline Build:
Input root, xData, xMap, yData, currentDepth, maxDepth
Output nodeRef

#Base Cases
if number of columns of xData == 1
make node from that column
key will be a pair [original column number, new column number]
class is which yData element is most likely
edge will be None
return that nodeRef
if currentDepth == maxDepth
make node
key will be None
class is which yData element is most likely
edge will be None
return that nodeRef

#Calculate the entropy of our attributes
for all col in xData:
Calculate entropy of xData[col] -> xEntropy[col]

#Calculate the estimate change in entropy for our attribute
#This is where we differ from ID3 to make simple
find col with minimal Entropy -> col

#make new node based on col minimal Entropy
make node for that column ->nodeRef
key will be a pair [original column number, new column number]
class is which yData element is most likely

#split xData and yData based on col
split on xData and yData based on col -> yDataList, xDataList

#loop over all lists and store in edges
for j in len(xDataList)
build(xDataList[j], newxMap, ...) -> returnedRef
noRef.edge{unique element, returnedRef}

return nodeRef

Task 4 Walk Algorithm.
After you build a tree in the train interface, you may want to make a prediction with eval. In order to make a prediction you will have to walk the tree based on the information in the the

xData. That is look up what values are in the columns to decide which children edges to follow until we get to a leaf node. At the leaf node, you will return the predicted class.

Task 5 Add Option to Menu.
Add an option to the menu that will allow you to pick if you use scikit-learn or your own tree.