Assignment 1: Decision Trees and Rule Induction
Mikhail Skobov
CSEP 546
Code: https://github.com/Skobovm/csep546/tree/assignment1

1.1 Write a paragraph describing how your code works.

The code is implemented exactly how it is described in Mitchell 3.4. First, the ARFF data is loaded via the scipy.io module. This data is used in the constructor of the initial node. When this root node is created, it is then determined whether or not all examples fall under the same label. If they do, the label is set to that value and the constructor returns. If attributes list is empty (no more attributes to choose from) the most common value is chosen. Otherwise, the constructor proceeds to choosing the best attribute to branch on. Once this is chosen, it is decided whether or not to actually branch any further. If not, the node label gets the most common target value of the examples at this point. Otherwise it goes on to segment the examples by their attribute value and then create new nodes recursively, passing in only the examples set where the best attribute value matches that attribute of the example. In order to have more testable code, most helper functions are static, and do not rely on class variables.

Note that this is not an optimal implementation and that there is a lot of redundant processing. This is an artifact of following the ID3 pseudocode from Mitchell. The recursive nature of this problem makes a recursive algorithm easier to implement, even though it can be unbearably slow. If I were to start again from the beginning, I would have implemented the tree creation in an iterative fashion.

1.2 – Training Data

Confidence	Accuracy (%)	Precision	Recall	Decision Nodes
0%	81.3	373/424	373/7807	1631
50%	80.7	213/320	213/7807	283
80%	81.1	349/460	349/7807	164
95%	80.5	58/79	58/7807	21
99%	80.6	57/82	57/7807	19

1.3 – Test Data

Confidence	Accuracy	Precision	Recall	Decision Nodes
0%	74.8	187/382	187/6292	1631
50%	75.0	177/309	177/6292	283
80%	75.6	366/548	366/6292	164
95%	74.8	12/20	12/6292	21
99%	74.9	16/22	16/6292	19

1.5 Which level of confidence produces the best results and why? What is your criterion for "the best results"?
While there does not seem to be a very large deviation between the results, it appears that the 80% confidence level produces the best results. In this case, best results refers to the highest overall accuracy on the test data. It is also second best when ran across the training data, and produces some of the higher precision and recall values for both sets. The most likely explanation for this is that 80% is the optimal confidence level for which the tree built by training data most accurately represents the set of test data. Higher confidence does not build out a large enough tree, while being less confident ends up overfitting.

1.6 Try to interpret the tree with the highest accuracy (on the test set). What are the first 4 decisions (or less if the tree has less than 4 levels) applied to the largest fraction of the positively-labeled training examples? Negatively-labeled? Express these paths as English sentences. Do they make sense?
Attribute: HowDidYouFindUs; Value: b'Friend/Co-worker'
Attribute: WhoMakesPurchasesForYou; Value: b'spouse'
Attribute: Num Wax Product Views; Value: b'"\\'(-inf-0.5]\\""
Attribute: Registration Gender; Value: b'Male'
Prediction: b'True'; Actual: b'True'

English: This person was referred to the site by a friend, his spouse makes purchases for him, he has viewed no wax products, and is male. What's interesting about this is that this path represents a large part of the data set. Upon printing out this data, it appears that the information from those variables is missing more often than not. I believe that this artifact is caused by blindly picking the mode of the attribute and that value becoming the "best" attribute for predicting a split in data.

1.7 If all your accuracies are low, how have you tried to improve them and what do you suspect is failing?
All of the accuracies are falling in the 75-80% range, which seems reasonable.

Extra Credit: Try to improve the predictive accuracy you obtained above. There are many approaches you could try. You could pick the best attribute to split on or handle unknown values differently. You could construct new attributes from the existing ones and augment the examples with them. You could try alternative classification techniques, or modify one that you used above. You could also go to the raw clickstream data provided on the KDD Cup website (this is separate from the data aggregated by session) and create new attributes directly from it. Note that this dataset is very large. A bunch of potentially useful pointers and free software can be found at KDnuggets.
[1] J. R. Quinlan. "Induction of decision trees". Machine Learning 1 (1) (1986): 81-106.

Problem 2: Rule Induction

2.1

a. A rule can be looked at as a boolean function taking n distinct inputs. Since a decision tree's "classification" function is the same as evaluating d inputs (if d is the tree depth) in order to get to a leaf node "label", we can say that there would be as many rules as there are leaf nodes (assuming a full tree). Thus,

$$|\text{rules}| = 2^{(d-1)}$$

b. Assuming that "precondition" is referring to the number of tests a rule has to make in order to make a decision, that number will be d – 1, where d is the depth of the tree

c. Sequential covering algorithm rules represent reaching decision tree leaf nodes (at least in this case). Thus, there must be as many rules as there are leaf nodes. Since the pseudocode for a sequential covering algorithms is one while loop that creates 1 rule with each iteration, this loop will go through the same number of iterations as there are rules times the number of attributes. The sequential covering algorithm will make $(2^{(d-1)}) * n$ "choices" for determining the same set of rules.

d. I would suspect that the sequential covering algorithm may be more prone to overfitting than a decision tree algorithm like ID3. I believe this is the case because these algorithms would over-fit entire truth equations, thus eliminating legitimate deviations from test data. Decision trees, on the other hand, could correct for this by split stopping or pruning at the individual attribute level.

2.2 Suppose FOIL is considering adding a literal to a clause using a binary predicate P.

a. Since it is a binary predicate, two new variables in a predicate is not allowed, and the negation of a predicate is considered to be a unique predicate, the following represents the number of distinct literals where n is the number of existing variables (subtract 1 for the case where it's the new variable as both inputs to the predicate)

$$2 * ((n + 1)^2 - 1)$$

b. I believe that FOIL does not allow literals that contain only new variables because that could create disjoint rules which do not relate back to the initial target. Using the Granddaughter example from the book, if you have Father(u, v) /\ Father(y,z) /\ Female(y) this does not give you any useful information relating back to y