CS-349 Homework #1

1. (0.25 points) Did you alter the Node data structure? If so, how and why?

Yes, we added a feature property and a leaf property.

- ***feature*** is the name of the attribute that the children of the current node are split based on. This is used for accessing the correct attribute values when evaluating examples.
- ***leaf*** is a boolean that indicates whether the node has children. This is to prevent falling out of the tree during evaluation.

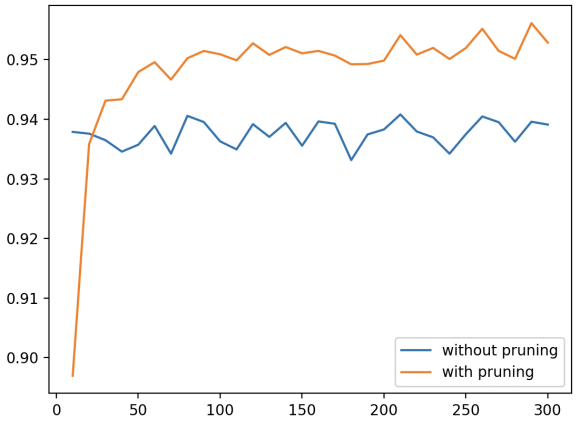
2. (0.25 points) How did you handle missing attributes, and why did you choose this strategy?

We handle missing attributes in the evaluate( ) function by returning the classification of our current node. We chose this because if you encounter a missing value, you have no way to descend the tree any further, making the current node the best classification you can make given the information at hand.

3. (0.5 points) How did you perform pruning, and why did you choose this strategy?

We performed error-complexity pruning by recursively removing subtrees from the bottom-up, calculating the pruned tree's accuracy on the validation data, and keeping the tree pruned if the accuracy was equal to or higher than the previously unpruned tree. We tried a number of other pruning strategies such as critical value pruning and stopping at a maximum tree depth, but this proved to be the most effective method for improving the accuracy of the model, and required implementing fewer additional fields to node.py than other methods.

4. (2.0 points) Now you will try your learner on the house_votes_84.data, and plot learning curves. Specifically, you should experiment under two settings: with pruning, and without pruning. Use training set sizes ranging between 10 and 300 examples. For each training size you choose, perform 100 random runs, for each run testing on all examples not used for training (see testPruningOnHouseData from unit_tests.py for one example of this). Plot the average accuracy of the 100 runs as one point on a learning curve (x-axis = number of training examples, y-axis = accuracy on test data). Connect the points to show one line representing accuracy with pruning, the other without. Include your plot in your pdf, and answer two questions:



a. What is the general trend of both lines as training set size increases, and why does this make sense?
The without pruning line seems to remain relatively steady over time, with a few lower data points as the dataset size grows. This makes sense because as the dataset grows, it becomes more difficult to overfit and will likely account for more general trends in the data. As the size of the dataset increases, the accuracy with pruning increases quickly. Then, the line remains relatively steady over different data sizes afterwards. This makes sense because pruning can remove important parts of a decision tree that should not necessarily be removed when the sample size is small, but as the sample size increases pruning helps expose more general trends in the training data that might have otherwise been missed and cut back on overfitting.

b. How does the advantage of pruning change as the data set size increases? Does this make sense, and why or why not? Note: depending on your particular approach, pruning may not improve accuracy consistently or may decrease it (especially for small data set sizes). You can still receive full credit for this as long as your approach is reasonable and correctly implemented.
As the dataset size increases, the advantage of pruning initially grows quickly before maintaining an overall benefit with minor fluctuations as data sample size increases. This makes sense as at first pruning is more important, because smaller datasets are easier to overfit to. Then it makes sense that pruning benefit remains relatively steady over time, because as the data set grows, the pruned tree accounts for more general patterns in the data than an overfit tree would.

5. (1.0 points) Use your ID3 code to learn a decision tree on cars_train.data. Report accuracy on the cars_train.data, cars_valid.data and cars_test.data datasets. If your accuracy on cars_train.data is less than 100%, explain how this can happen. Prune the decision tree learned on cars_train.data using cars_valid.data. Run your pruned decision tree on cars_test.data, and explain the resulting accuracy on all three datasets.

Training data accuracy: 1.0
Testing data accuracy: 0.7142857142857143
Validation data accuracy: 0.8

Training accuracy after pruning: 0.815
Testing accuracy after pruning: 0.7714285714285715
Validation accuracy after pruning: 0.8

The training data accuracy reduces because the tree is no longer specifically fit to the training data after pruning for general improvements. The testing data accuracy improves because the pruning generalizes the data better for datasets that the model is not trained on. The validation accuracy stays the same because either the pruning worsened the validation data and did not occur, or the pruning that did occur did not affect the validation data points.

6. (2.0 points) Use your ID3 code to construct a Random Forest classifier using the candy.data dataset. You can construct any number of random trees using methods of your choosing. Justify your design choices and compare results to a single decision tree constructed using the ID3 algorithm.

We used 20 trees for our random forest and 30 samples with replacement from the training set to train each tree. We used 20 trees because it seems to be a sufficient number of trees for the size of our dataset, and any more would give redundant results without a significant increase in accuracy. We used 30 samples because we believed that it would be enough data for a given decision tree to learn significant patterns in our dataset, while also not having too much data to the point where each tree in the forest would essentially be the same, considering that candy.data only had ~85 examples and those had to be split up for training, validation and testing. Below are the accuracy results of the random forest and a single decision tree on the candy dataset. It is worth noting that while the single tree and random forest yield similar results on the candy dataset, the random forest consistently outperformed the single tree on larger datasets like the house votes data, and we provide accuracy results for that down below as well.

Candy Dataset Accuracy Results

	Training	Validation	Test
Single Tree	85.71%	57.14%	72.72%
Random Forest	85.71%	76.19%	72.72%

Housing Vote Dataset Accuracy Results

	Training	Validation	Test
Single Tree	100%	99.08%	90.83%
Random Forest	96.31%	98.17%	95.41%