Gale Curry

Questions

1. ***How did you handle missing attributes in examples?***
   During training, every time before I split on an attribute, I checked if there were any missing values for that attribute. If there were, and it was a **numerical attribute**, I calculated the average value of the data set for that attribute and set the missing value equal to this average value*. If the missing value was **nominal**, I left it as None. This allows examples to be sorted into regular values like 1, 2, 3, as well as None. So during validation or prediction, if an example is missing that attribute as well it will be sorted into None. This is like treating the missing attribute as just another value the attribute could have.
   *When making these changes I altered a "deep" copy of the data set.

2. ***Apply your algorithm to the training set, without pruning. Print out a Boolean formula in disjunctive normal form that corresponds to the unpruned tree learned from the training set. For the DNF assume that group label "1" refers to the positive examples. NOTE: if you find your tree is cumbersome to print in full, you may restrict your print-out to only 16 leaf nodes.***

   For my DNF function I implemented it so that it would print to the screen (not output to a text file). I also chose to include the paths which led to negative examples (winner = 0), so I could see better all of the outcomes. It is not as succinct as the traditional DNF format but it is easier to read in my opinion. I also assumed that "your team" is Team "1" aka when winner = 1 that's Team 1 winning. Here is a snippet of my output (16 nodes):

```
IF
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
oppnuminjured is less than 2.0...AND...
opprundifferential is greater than 40.0 ...AND...
dayssincegame is greater than 1.0 ...AND...
oppstartingpitcher is key 1...AND...
weather is key 0.490530646...AND...
oppwinningpercent is greater than 0.7146318 ...AND...
Your team loses.
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
oppnuminjured is less than 2.0...AND...
opprundifferential is greater than 40.0 ...AND...
dayssincegame is greater than 1.0 ...AND...
oppstartingpitcher is key 1...AND...
weather is key 0.490530646...AND...
Your team loses.
OR
```

numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
oppnuminjured is less than 2.0...AND...
opprundifferential is greater than 40.0 ...AND...
dayssincegame is greater than 1.0 ...AND...
oppstartingpitcher is key 1...AND...
startingpitcher is key 0.388647802...AND...
Your team wins!
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
oppnuminjured is less than 2.0...AND...
opprundifferential is greater than 40.0 ...AND...
dayssincegame is greater than 1.0 ...AND...
oppstartingpitcher is key 1...AND...
startingpitcher is key 0.388647802...AND...
Your team loses.
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
oppnuminjured is less than 2.0...AND...
opprundifferential is greater than 40.0 ...AND...
dayssincegame is greater than 1.0 ...AND...
oppstartingpitcher is key 1...AND...
startingpitcher is key 0.388647802...AND...
Your team loses.
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
winpercent is greater than 0.543700373 ...AND...
oppnuminjured is less than 3.0...AND...
oppnuminjured is less than 2.0...AND...
oppwinningpercent is less than 0.648694639...AND...
Your team wins!
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
winpercent is greater than 0.543700373 ...AND...
oppnuminjured is less than 3.0...AND...
oppnuminjured is less than 2.0...AND...
oppwinningpercent is less than 0.648694639...AND...

```
rundifferential is greater than 47.0 ...AND...
oppstartingpitcher is key 0...AND...
oppnuminjured is greater than 1.53333333333 ...AND...
Your team wins!
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
winpercent is greater than 0.543700373 ...AND...
oppnuminjured is less than 3.0...AND...
oppnuminjured is less than 2.0...AND...
oppwinningpercent is less than 0.648694639...AND...
rundifferential is greater than 47.0 ...AND...
oppstartingpitcher is key 0...AND...
oppnuminjured is greater than 1.53333333333 ...AND...
Your team loses.
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
winpercent is greater than 0.543700373 ...AND...
oppnuminjured is less than 3.0...AND...
oppnuminjured is less than 2.0...AND...
oppwinningpercent is less than 0.648694639...AND...
rundifferential is greater than 47.0 ...AND...
oppstartingpitcher is key 0...AND...
Your team wins!
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
winpercent is greater than 0.543700373 ...AND...
oppnuminjured is less than 3.0...AND...
oppnuminjured is less than 2.0...AND...
oppwinningpercent is less than 0.648694639...AND...
rundifferential is greater than 47.0 ...AND...
oppstartingpitcher is key 0...AND...
Your team wins!
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
winpercent is greater than 0.543700373 ...AND...
oppnuminjured is less than 3.0...AND...
oppnuminjured is less than 2.0...AND...
oppwinningpercent is less than 0.648694639...AND...
```

```
rundifferential is greater than 47.0 ...AND...
oppstartingpitcher is key 0...AND...
Your team wins!
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
winpercent is greater than 0.543700373 ...AND...
oppnuminjured is less than 3.0...AND...
oppnuminjured is less than 2.0...AND...
oppwinningpercent is less than 0.648694639...AND...
rundifferential is greater than 47.0 ...AND...
oppstartingpitcher is key 0...AND...
Your team loses.
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
winpercent is greater than 0.543700373 ...AND...
oppnuminjured is less than 3.0...AND...
oppnuminjured is less than 2.0...AND...
oppwinningpercent is less than 0.648694639...AND...
rundifferential is greater than 47.0 ...AND...
Your team wins!
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
winpercent is greater than 0.543700373 ...AND...
oppnuminjured is less than 3.0...AND...
oppnuminjured is less than 2.0...AND...
opprundifferential is greater than 58.0 ...AND...
Your team loses.
OR
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
winpercent is greater than 0.543700373 ...AND...
oppnuminjured is less than 3.0...AND...
oppnuminjured is less than 2.0...AND...
opprundifferential is greater than 58.0 ...AND...
temperature is greater than 72.60861278 ...AND...
homeaway is key 0...AND...
winpercent is greater than 0.157968817 ...AND...
Your team wins!
OR
```

```
numinjured is less than 2.0
numinjured is less than 1.0...AND...
opprundifferential is greater than 53.0 ...AND...
winpercent is greater than 0.543700373 ...AND...
oppnuminjured is less than 3.0...AND...
oppnuminjured is less than 2.0...AND...
opprundifferential is greater than 58.0 ...AND...
temperature is greater than 72.60861278 ...AND...
homeaway is key 0...AND...
winpercent is greater than 0.157968817 ...AND...
        Your team loses.
```

3. ***Explain in English one of the rules in this (unpruned) tree.***
   My rules are already stated in (mostly) plain English but here is an example taken from the last rule printed in the previous question:
   If the number of injured players on one team is less than 2.0and the number of injured players on the other team is less than 1.0 and the run differential for the opposing team is greater than 53.0 and the win percentage is greater than 0.54 and the number of injured players on the opposing team is less than 3.0 and the number of injured players on the opposing team is less than 2.0 and the opposing team's run differential is greater than 58.0 and the temperature is greater than 72.6 degrees and it is a home game and the win percentage is greater than 0.15 then… Your team (Team 1) loses.

4. ***How did you implement pruning?***
   I first use a function to "traverse" through all of the branches of the tree (using recursion). At the end of each branch, I store that end node's parent, and the output of my traverse function is an array of all nodes that I may want to trim. I then test the effect of deleting each of these nodes, effectively trimming the outermost leaves of the tree. To delete a node, I make its parent node output a 1 or 0. If that trim results in a better validation accuracy than the previous best value, I do not restore the node. If the accuracy is lower, I "restore" the node. I can do this because every time I deleted a node, I made a deep copy of the original before I made any changes. I restore the node with its original values, as well as update a flag called *node.pruned* to True to indicate that I have already attempted to prune that leaf. After trying to trim all of the outer leaves, I may have kept some changes and therefore there are new leaves to try trimming. I call the traverse function again and get the updated list of outermost leaves. I trim these as well and continue on calling traverse until further loops stop resulting in better accuracy.
   I know that only trimming the outer leaves is a conservative way to implement pruning. Since my accuracy still increases with this kind of pruning, I am happy with it.

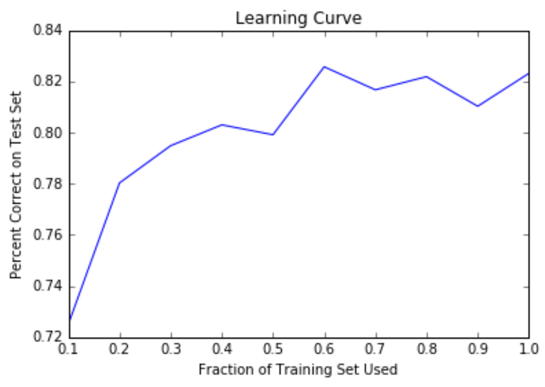
5. ***Apply your algorithm to the training set, with pruning. Print out a Boolean formula in disjunctive normal form that corresponds to the pruned tree learned from the training set.***
   I used the large data set but with depth = 10 and it still couldn't all fit here. Sorry! See code for my imperfect implementation of DNF.

6. **What is the difference in size (number of splits) between the pruned and unpruned trees?**
Unpruned Splits: = 296, Pruned Splits: 290
There were only 6 prunes that were kept. This is again because I was only pruning the leaves of the tree.

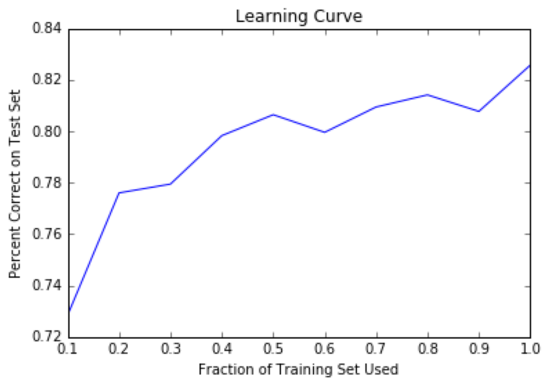
7. ***Test the unpruned and pruned trees on the validation set. What are the accuracies of each tree? Explain the difference, if any.***
Conditions: Using the large training set, depth and numerical_splits_count both set to 30, steps = 5000.
Unpruned accuracy: 83.4%
Pruned accuracy: 83.75%
The pruned accuracy is higher because it was able to make a few small prunes. As I mentioned above, my pruning is very safe, so the difference is not very significant between the two.

8. **Create learning curve graphs for both unpruned and pruned trees. Is there a difference between the two graphs?**

These are the learning curves for unpruned and pruned on the small data set with steps= 10. These were tested on the validation set, test_bvalidate.csv. It was not clear if the learning curve should be validated on the remaining portion of the training set, like in cross-validation, so I tested on the given validation set.



Unpruned



Pruned

The difference between the curves is that the learning curve is steeper for the unpruned tree.

9. **Which tree do you think will perform better on the unlabeled test set? Why? Run this tree on the test file and submit your predictions as described in the submission instructions.**

I think the pruned tree will run better on the unlabeled test set. This is because the unpruned tree is more likely to be over-fit to the training data. A tree which has been pruned is less likely to mis-classify unseen examples due to noise in the training set.
Conditions: Using the large training set, with pruning, depth and numerical_splits_count both set to 30, steps = 5000.

10. **Which members of the group worked on which parts of the assignment?**

    Roman and I worked together to help each other think through ideas as well as debugging. We wrote the entirety of our code separately.

11. **BONUS: This assignment used Information Gain Ratio instead of Information Gain (IG) to pick attributes to split on, which is expected to boost accuracy over IG. We also used a limited step side for numeric attributes instead of testing all possible attributes as split points. Were these good model selections? Try using plain IG and see if this impacts validation set accuracy. Likewise, try testing all numeric split points (doing so efficiently will probably require writing new code, rather than just setting steps = 1), and evaluate whether this improves validation set accuracy.**

    I implemented a module called ID3_plainIG in which my ID3 function ran using Information Gain instead of Information Gain Ratio. I ran the same data set through both ID3 modules and compared their accuracy on the validation set. These were my settings for both runs:

    Training Set: test_btrain.csv (small data set)
    Validation Set: test_bvalidate.csv
    Pruning: False
    Splits_on_numerical: 30
    Depth: 30
    Steps: 10

    And my results were as follows:

    Accuracy with Information Gain Ratio: 85.0%
    Accuracy with Information Gain: 82.2%

    For the small data set (test_btrain.csv) I also tried testing all numeric split points, by setting steps= 1 (since on the small set it didn't take too long). I used the same settings as above except with steps= 1.

    Accuracy with Information Gain Ratio: 86.5%
    Accuracy with Information Gain: 83.9%

    We can say that increasing steps increased accuracy for both cases. But the increase is not by much. The difference in accuracy between using IG and IGR are significant enough to say that implementing IGR was the right choice, especially since there is no noticeable difference in training time.