EECS 349 Machine Learning
Problem Set 2

Team member: Feng Hong ; Zhao Xiong ; Ruiqi Yang
Netid: fho400 ; zxe253 ; rya743

1. How did you represent the decision tree in your code?

We use a dictionary to save our decision tree. For every node, the attribute will be stored as the key in the dictionary, and all the information, including the judging condition and the leaf information, will be stored as the values. Below is part of our decision tree.

```
{'numinjured': {'>1.0': {'oppnuminjured': {'<=2.0':
{'rundifferential': {'>42.0': {'opprundifferential': {'<=33.0':
{'opprundifferential': {'>1.0': {'opprundifferential': {'>9.0':
{'homeaway': {'<=0.0': {'rundifferential': {'>59.0':
{'opprundifferential': {'>26.0': {'weather': {'<=0.0':
{'rundifferential': {'>79.0': 1.0, '<=79.0': {'oppnuminjured':
{'<=0.0': 1.0, '>0.0': {'oppwinpercent': {'<=0.3995035911':
{'startingpitcher': {'<=2.0': 1.0, '>2.0': {'dayssincegame':
{'>3.0': 0.0, '<=3.0': {'rundifferential': {'<=66.0':
{'weather': {'<=-1.0': 0.0, '>-1.0': 1.0}
```

Briefly, ‘numinjured’ is the first node in this tree, and the judging condition will be ‘>1.0’ or ‘<=1.0’. If the value of ‘numinjured’ in an example is more than 1.0, it will continue to detect its ‘oppnuminjured’ value. This process will continue until there is only ‘1.0’ or/and ‘0.0’ after the judging condition.

2. How did you represent examples (instances) in your code?

The first thing to deal with the examples is to convert the cvs file to the string in python. We use ‘,’ as the delimiter to distinguish every value, and store it into a two-dimension list. Then, we fix the missing data problem (see Q4), and convert all string to float. That’s how we get a complete and comparable example list.

3. How did you choose the attribute for each node?

Generally, we calculate the entropy gain of each feature. For each attribute, we try every existed value in this column of the current dataset as a threshold, set it as the judging condition, calculate the entropy gain of this value, and choose the biggest one. Then choose attribute with the biggest entropy gain as our feature for the current node.

4. How did you handle missing attributes in examples?

First of all, we delete the examples which have unknown results(missing the ***winner*** value). Then, we classify our attributes, except ***winner***, into two categories, Numeric and Categorical. We use List to record every existed value in each column, and the value will be the times this value exits. If the length of list is more than 100, which means there are more than 100 different values in this column, we assume it is Numeric, otherwise, we assume it is Categorical. For the Numeric attribute, we use the average value of existed values to replace the missing data, and for the Categorical attribute, we use the mode (the most frequent one) to replace the missing data.

5. What is the termination criterion for your learning process?

We have two termination criterion in our code. The first one is if there is no entropy gain in current dataset, we stop and use the majority result as the output. The second one is if there is only one example left in the dataset, we stop and use the result of this example as the output.

6. 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.

(part of the unpruned tree)

numinjured >1.0
- oppnuminjured <=2.0
-- rundifferential >42.0
--- opprundifferential <=33.0
---- opprundifferential >1.0
----- opprundifferential >9.0
------ homeaway <=0.0
------- rundifferential >59.0
-------- opprundifferential >26.0
--------- weather <=0.0
---------- 1.0
--------- weather >0.0
---------- 1.0
-------- opprundifferential <=26.0
--------- opprundifferential <=25.0
---------- rundifferential >82.0

---------- 1.0
---------- rundifferential <=82.0
---------- rundifferential >81.0
---------- 1.0
---------- rundifferential <=81.0
---------- 1.0
--------- opprundifferential >25.0
---------- oppstartingpitcher >1.0
---------- startingpitcher <=2.0
……
--------- rundifferential <=46.0
---------- oppstartingpitcher <=2.0
---------- opprundifferential >35.0
---------- homeaway <=0.0
---------- rundifferential >14.0
---------- oppnuminjured <=2.0
---------- rundifferential <=31.0
---------- startingpitcher >4.0
---------- 1.0
---------- startingpitcher <=4.0
---------- 0.0
---------- rundifferential >31.0
---------- temperature <=64.8798695053
---------- oppstartingpitcher >1.0
---------- 0.0
---------- oppstartingpitcher <=1.0
---------- 0.0
---------- temperature >64.8798695053
---------- 0.0
---------- oppnuminjured >2.0
---------- 0.0

7. Explain in English one of the rules in this (unpruned) tree.

If the value of ***rundifferential*** of this example is less than 46.0, then we go check ***oppstartingpitcher*** value. If it is less than 2.0, we go continue to check the value of ***opprundifferential***, ***homeaway***, ***rundifferential***, and ***oppnuminjured***. Then, we check the value of ***rundifferential***, if it is less than 31.0, we go check ***startingpitcher***, or we go check the value of ***temperature***. If in this case, the value of ***rundifferential*** is less than 31, and the value of ***startingpitcher*** is more then 4.0, we assume the output will be positive. If the value of ***startingpitcher*** is less then or equal to 4.0, we assume the output is negative.

8. How did you implement pruning?

If less than 50 examples in this node, we choose the majority result to replace this node.

Attempt: We try to use DFS to traverse all the nodes and testing whether delete this node will perform better than remain it. While there always come up with some I/O problems.
Well, we believe this process can be similar to testing the example numbers left.

9. 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.

(part of the tree)
numinjured >1.0
-oppnuminjured <=2.0
--rundifferential >42.0
---opprundifferential <=33.0
----opprundifferential >1.0
-----opprundifferential >9.0
------homeaway <=0.0
-------rundifferential >59.0
--------opprundifferential >26.0
---------weather <=0.0
----------1.0
---------weather >0.0
----------1.0
--------opprundifferential <=26.0
---------opprundifferential <=25.0
----------1.0 (different from here)
---------opprundifferential >25.0
----------1.0
-------rundifferential <=59.0
--------startingpitcher <=2.0
---------oppwinpercent <=0.511843921816
----------winpercent <=0.801354409958
-----------1.0
----------winpercent >0.801354409958
-----------1.0
---------oppwinpercent >0.511843921816
----------1.0
--------startingpitcher >2.0

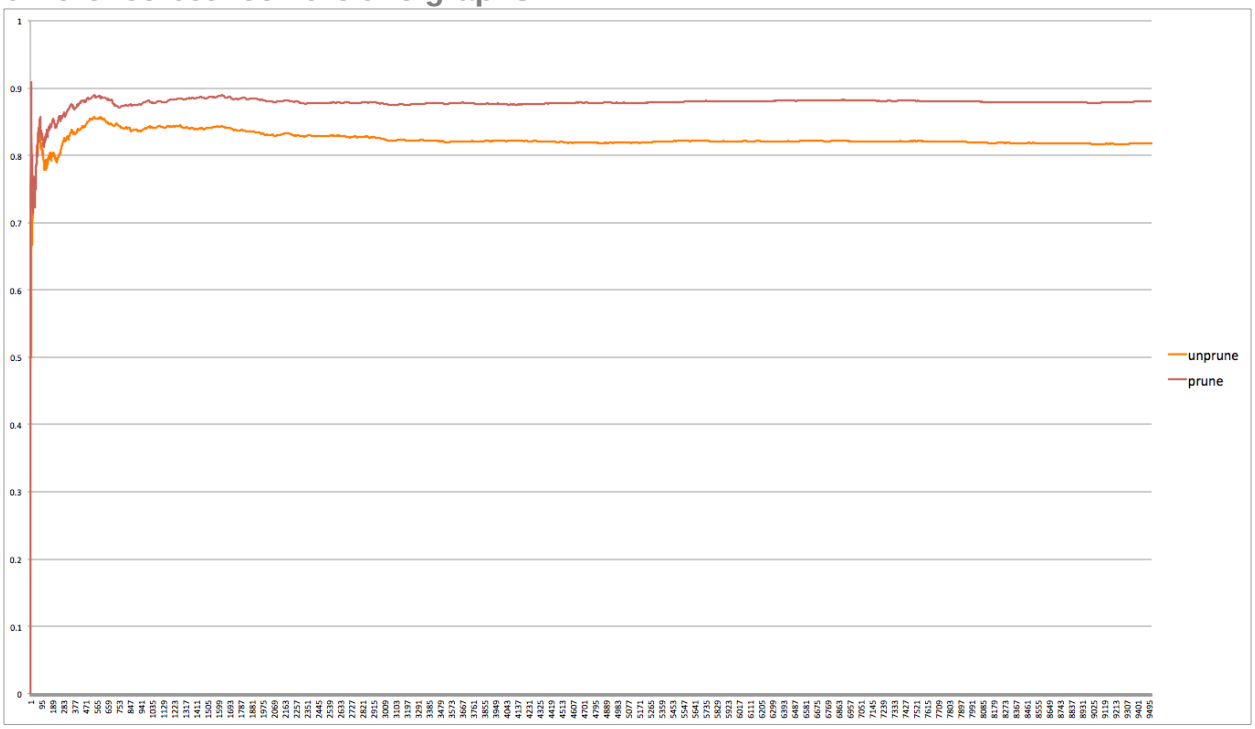
10. What is the difference in size (number of splits) between the pruned and unpruned trees?

pruned tree: 2774
unpruned tree: 9795
Apparently, pruned tree has a less tree size.

11. Test the unpruned and pruned trees on the validation set. What are the accuracies of each tree? Explain the difference, if any.

The accuracy of the unpruned one is 0.817981304485.
The accuracy of the pruned one is 0.88026467808.
The pruned one avoids overfitting problem, so the accuracy is better than the unpruned one.

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



As the graph above, these two learning curve follow a similar pattern. Firstly increase sharply, then some fluctuating, and when there are enough examples, the accuracy tents to keep a constant value. The only difference is that pruned one has a better accuracy than unpruned one.

13. 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.

The pruned on will perform better. It replies less on the training data, and avoids the overfitting problem.

14. What aspects of the feature set (if any) are a good fit for decision trees, and what aspects aren't a good fit?

good: wining percentage related, and numbers of injured player
not good: meway, dayssincegame

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

Zhao Xiong: Data Processing / Tree Drawing / Learning Curve Drawing
Feng Hong: Decision Tree
Ruiqi Yang: Test Function
All the brainstorm are finished by all three persons.