EECS349 Machine learning PS2
Weihan Chu (wcm350)

1.How did you handle missing attributes in examples?

As for the missing attributes, there are two major different case:

If the missing attributes are in the train-set or in the validation-set, if the missing attribute is the 'winner' attribute, we can just delete this row, because this data has no sense. If the missing attributes are in other attributes, we need to consider them on their data-type, if they are nominal, we will pick the mode data from all non-none data to represent this missing attribute. If they are numeric, we just pick the average from all non-none data to represent this missing attribute.

If the missing attributes are in the predict-set, all the 'winner attribute' has no value. We don't need to check the 'winner' attribute. We just check other attributes, if they are nominal, we will pick the mode data from all non-none data to represent this missing attribute. If they are numeric, we just pick the average from all non-none data to represent this missing attribute.

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.

we use :
steps:1
limit_splits_on_numerical: 5
limit_depth: 10

(numinjured<1.0 ^ opprundifferential<13.0) v (numinjured<1.0 ^ opprundifferential>=13.0 ^ oppnuminjured<0.0) v (numinjured<1.0 ^ opprundifferential>=13.0 ^ oppnuminjured>=0.0 ^ oppnuminjured<1.0 ^ oppwinningpercent>=0.381126765 ^ numinjured>=0.0 ^ numinjured<0.913173652695 ^ temperature<66.72394947) v (numinjured<1.0 ^ opprundifferential>=13.0 ^ oppnuminjured>=0.0 ^ oppnuminjured<1.0 ^ oppwinningpercent>=0.381126765 ^ numinjured>=0.0 ^ numinjured<0.913173652695 ^ temperature>=66.72394947 ^ opprundifferential<70.0) v (numinjured<1.0 ^ opprundifferential>=13.0 ^ oppnuminjured>=0.0 ^ oppnuminjured>=1.0 ^ rundifferential<2.0) v (numinjured<1.0 ^ opprundifferential>=13.0 ^ oppnuminjured>=0.0 ^ oppnuminjured>=1.0 ^ rundifferential>=2.0 ^ opprundifferential<25.0 ^ rundifferential>=66.0) v (numinjured<1.0 ^ opprundifferential>=13.0 ^ oppnuminjured>=0.0 ^ oppnuminjured>=1.0 ^ rundifferential>=2.0 ^ opprundifferential>=25.0 ^ rundifferential>=82.0 ^ opprundifferential<51.0) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^ temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^ oppnuminjured<2.0 ^ oppwinningpercent<0.195529965 ^ oppwinningpercent<0.050022726) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^ temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^

oppnuminjured<2.0 ^ oppwinningpercent<0.195529965 ^
oppwinningpercent>=0.050022726 ^ winpercent<0.317461972) v (numinjured>=1.0 ^
oppnuminjured<4.0 ^ temperature>=48.44667111 ^ opprundifferential<106.0 ^
rundifferential<120.0 ^ oppnuminjured<2.0 ^ oppwinningpercent>=0.195529965 ^
winpercent<0.929746511 ^ oppwinningpercent<0.208758923 ^
temperature>=64.60280261) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^
temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^
oppnuminjured<2.0 ^ oppwinningpercent>=0.195529965 ^ winpercent<0.929746511 ^
oppwinningpercent>=0.208758923 ^ winpercent<0.901611692) v (numinjured>=1.0 ^
oppnuminjured<4.0 ^ temperature>=48.44667111 ^ opprundifferential<106.0 ^
rundifferential<120.0 ^ oppnuminjured<2.0 ^ oppwinningpercent>=0.195529965 ^
winpercent<0.929746511 ^ oppwinningpercent>=0.208758923 ^
winpercent>=0.901611692) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^
temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^
oppnuminjured<2.0 ^ oppwinningpercent>=0.195529965 ^ winpercent>=0.929746511 ^
dayssincegame>=4.0) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^
temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^
oppnuminjured>=2.0 ^ numinjured<1.91949910555 ^ rundifferential<95.0 ^
oppwinningpercent<0.061920854) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^
temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^
oppnuminjured>=2.0 ^ numinjured<1.91949910555 ^ rundifferential<95.0 ^
oppwinningpercent>=0.061920854 ^ oppdayssincegame<1.0) v (numinjured>=1.0 ^
oppnuminjured<4.0 ^ temperature>=48.44667111 ^ opprundifferential<106.0 ^
rundifferential<120.0 ^ oppnuminjured>=2.0 ^ numinjured<1.91949910555 ^
rundifferential>=95.0) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^
temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^
oppnuminjured>=2.0 ^ numinjured>=1.91949910555 ^ dayssincegame>=0.0 ^
oppnuminjured<2.04451038576 ^ temperature<81.50095868) v (numinjured>=1.0 ^
oppnuminjured<4.0 ^ temperature>=48.44667111 ^ opprundifferential<106.0 ^
rundifferential<120.0 ^ oppnuminjured>=2.0 ^ numinjured>=1.91949910555 ^
dayssincegame>=0.0 ^ oppnuminjured<2.04451038576 ^ temperature>=81.50095868)
v (numinjured>=1.0 ^ oppnuminjured<4.0 ^ temperature>=48.44667111 ^
opprundifferential<106.0 ^ rundifferential<120.0 ^ oppnuminjured>=2.0 ^
numinjured>=1.91949910555 ^ dayssincegame>=0.0 ^
oppnuminjured>=2.04451038576 ^ numinjured>=3.0) v (numinjured>=1.0 ^
oppnuminjured>=4.0 ^ temperature<62.69854496 ^ oppdayssincegame>=3.0)

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

According to the tree, we know that the root node is number of injured players on one team.That is to say, the most significant attribute to decide whether a team win or not is the number of injured players. Actually , this result accords with our common sense. Compared with other attributes such as temperature, the # of people who got injured counts much more here.

What's more,

numinjured >= 1.0

oppnuminjured >= 4.0

temperature >= 62.69854496 : 0

this means when number of injured players is more than 1, the opponent team's injured players is more than 4 , the temperature is more than 62.698, then the label is 0

4.How did you implement pruning?

We adopted a top-down pruning strategy here. If one of the children of the node we currently deal with is labeled as None, which means it is an interior node, we will prune it based on the result of comparison between the validation accuracy before pruning and after pruning. Then, compare the accuracy of the pruned tree with the unpruned tree, if the accuracy is improved, then prune this child. Else, don't prune this child, and try to prune this child's children recursively.

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.

we use :
steps:1
limit_splits_on_numerical: 5
limit_depth: 10

(numinjured<1.0 ^ opprundifferential<13.0) v (numinjured<1.0 ^ opprundifferential>=13.0 ^ oppnuminjured<0.0) v (numinjured<1.0 ^ opprundifferential>=13.0 ^ oppnuminjured>=0.0 ^ oppnuminjured<1.0) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^ temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^ oppnuminjured<2.0 ^ oppwinningpercent<0.195529965 ^ oppwinningpercent<0.050022726) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^ temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^ oppnuminjured<2.0 ^ oppwinningpercent>=0.195529965) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^ temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^ oppnuminjured>=2.0 ^ numinjured>=1.91949910555 ^ dayssincegame>=0.0 ^ oppnuminjured<2.04451038576) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^ temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^ oppnuminjured>=2.0 ^ numinjured>=1.91949910555 ^ dayssincegame>=0.0 ^ oppnuminjured>=2.04451038576 ^ numinjured>=3.0) v (numinjured>=1.0 ^ oppnuminjured>=4.0 ^ temperature<62.69854496 ^ oppdayssincegame>=3.0)

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

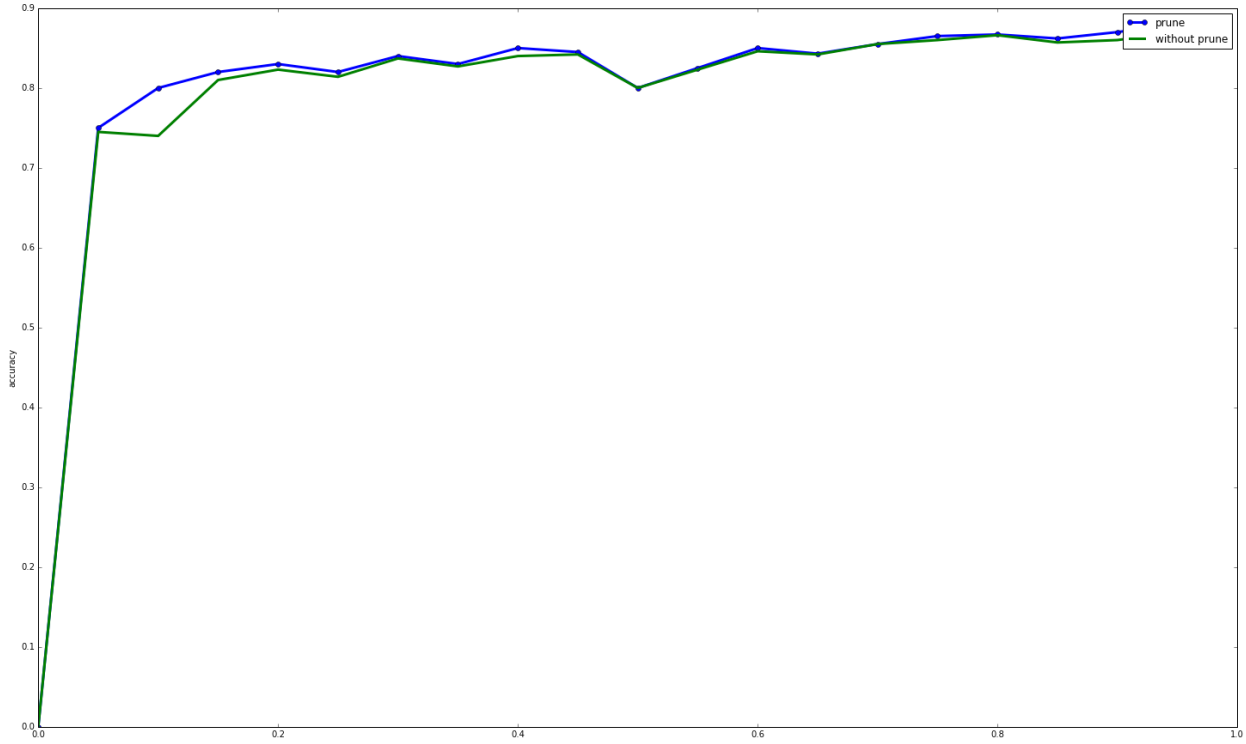
we use :
steps:1
limit_splits_on_numerical: 5
limit_depth: 10
The size of unpruned tree is 98 and the size of pruned tree is 54

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

we use :
steps:1
limit_splits_on_numerical: 5
limit_depth: 10

The accuracy before pruning is 87.3931623932, the one after pruning is 88.0341880342. Since we can eliminate some unnecessary branches to avoid overfitting problem, the accuracy on the validation data could be a little higher.

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



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 that the pruned tree will perform better on the unlabeled test dataset. Because it avoid the overfitting and then has a higher accuracy on the validation set, so the accuracy of the tree model through the pruned tree is better.

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

ID3:Weihan Chu and Zhaocheng Yu
Pruning: Cao Xue

Data Pre-processing: Weihan Chu
Print-tree:Zhaocheng Yu
Classify: Weihan Chu and Cao Xue
Learning Curve: Zhaocheng Yu and Cao Xue
And we work together for four days to debug to improve our model

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.

Yes, they both are very good model selections. Firstly, the information gain ratio is better than the information because the information gain ratio works better when there are a large number of distinct values.

Step	1	10	50	100
IG ratio	88.034	87.802	86.474	80.121
IG	86.764	85.987	84.963	79.234

when we use steps:1, limit_splits_on_numerical: 5,limit_depth: 10 for the pruned tree. If we use IG ratio, the accuracy will be 88.034, but when we use the IG, the accuracy will be 86.794%. The test result also prove our theory. So IG ratio is better than IG. And when we change the steps, the IG ratio's accuracy is always better than IG

For the steps, when we test the all numeric split points and when we increase the steps, the accuracy will decrease and the train time will also decrease.

some other information is in next page

Appendix

1.result for auto_grader.py:

```
========== homogenous ==========
Passed 1
Passed 2
Passed 3
All tests passed
========== pick_best_attribute ==========
Passed 1
Passed 2
All tests passed
========== mode ==========
Passed 1
Passed 2
All tests passed
========== entropy ==========
Passed 1
Passed 2
Passed 3
All tests passed
========== gain_ratio_nominal ==========
Passed 1
Passed 2
Passed 3
All tests passed
========== gain_ratio_numeric ==========
Passed 1
Passed 2
Passed 3
All tests passed
========== split_on_nominal ==========
Passed 1
Passed 2
All tests passed
========== split_on_numerical ==========
Passed 1
Passed 2
All tests passed
========== classify ==========
Passed 1
Passed 2
Passed 3
All tests passed
========== ID3 ==========
```

Passed 1
Passed 2
Passed 3
all tests passed.
Passed nominal test

2. result for decision_tree_driver.py

```
###
#  Training Tree
###


finished ID3
###
#  Validating
###
Accuracy before pruning: 87.6068376068

###
#  Pruning
###
87.6068376068
87.6068376068
87.8205128205
87.8205128205
88.0341880342
88.0341880342

###
#  Decision Tree
###
numinjured >= 1.0
  oppnuminjured >= 4.0
    temperature >= 62.69854496 : 0
    temperature < 62.69854496
      oppdayssincegame >= 3.0 : 1
      oppdayssincegame < 3.0 : 0
  oppnuminjured < 4.0
    temperature >= 48.44667111
      opprundifferential >= 106.0 : 0
      opprundifferential < 106.0
        rundifferential >= 120.0 : 0
        rundifferential < 120.0
          oppnuminjured >= 2.0
            numinjured >= 1.91949910555
```

```
            dayssincegame >= 0.0
              oppnuminjured >= 2.04451038576
                numinjured >= 3.0 : 1
                numinjured < 3.0 : 0
              oppnuminjured < 2.04451038576 : 1
            dayssincegame < 0.0 : 0
          numinjured < 1.91949910555 : 0
        oppnuminjured < 2.0
          oppwinningpercent >= 0.195529965 : 1
          oppwinningpercent < 0.195529965
            oppwinningpercent >= 0.050022726 : 0
            oppwinningpercent < 0.050022726 : 1
    temperature < 48.44667111 : 0
numinjured < 1.0
  opprundifferential >= 13.0
    oppnuminjured >= 0.0
      oppnuminjured >= 1.0 : 0
      oppnuminjured < 1.0 : 1
    oppnuminjured < 0.0 : 1
  opprundifferential < 13.0 : 1
Decision Tree written to /output/tree

###
#  Decision Tree as DNF
###
(numinjured<1.0 ^ opprundifferential<13.0) v (numinjured<1.0 ^
opprundifferential>=13.0 ^ oppnuminjured<0.0) v (numinjured<1.0 ^
opprundifferential>=13.0 ^ oppnuminjured>=0.0 ^ oppnuminjured<1.0) v
(numinjured>=1.0 ^ oppnuminjured<4.0 ^ temperature>=48.44667111 ^
opprundifferential<106.0 ^ rundifferential<120.0 ^ oppnuminjured<2.0 ^
oppwinningpercent<0.195529965 ^ oppwinningpercent<0.050022726) v
(numinjured>=1.0 ^ oppnuminjured<4.0 ^ temperature>=48.44667111 ^
opprundifferential<106.0 ^ rundifferential<120.0 ^ oppnuminjured<2.0 ^
oppwinningpercent>=0.195529965) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^
temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^
oppnuminjured>=2.0 ^ numinjured>=1.91949910555 ^ dayssincegame>=0.0 ^
oppnuminjured<2.04451038576) v (numinjured>=1.0 ^ oppnuminjured<4.0 ^
temperature>=48.44667111 ^ opprundifferential<106.0 ^ rundifferential<120.0 ^
oppnuminjured>=2.0 ^ numinjured>=1.91949910555 ^ dayssincegame>=0.0 ^
oppnuminjured>=2.04451038576 ^ numinjured>=3.0) v (numinjured>=1.0 ^
oppnuminjured>=4.0 ^ temperature<62.69854496 ^ oppdayssincegame>=3.0)
Decision Tree written to /output/DNF

###
#  Validating
###
```

Accuracy after pruning: 88.0341880342

###
Generating Predictions on Test Set
###