Jonathan Bohrer, Alex Gangwish
EECS 349
Problem Set 2 Answers

1. We handled missing attributes by calculating the mode of every attribute for the data set and storing them in an array. Then, we assigned the corresponding modal value to each missing attribute.

2. (restricted to 16 leaf nodes)
(oppnuminjured>=3.0 ^ opprundifferential<18.0 ^ rundifferential>=39.0 ^ opprundifferential>=16.0 ^ rundifferential>=42.0 ^ homeaway=0 ^ rundifferential<59.0 ^ winpercent<0.404196846425) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured<0.0 ^ oppnuminjured<0.0 ^ winpercent<0.590582828588 ^ oppdayssincegame>=2.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured<0.0 ^ oppnuminjured<0.0 ^ winpercent>=0.590582828588 ^ oppwinningpercent>=0.494600828567 ^ winpercent>=0.600132770207) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured<0.0 ^ oppnuminjured>=0.0 ^ opprundifferential<18.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured<0.0 ^ oppnuminjured>=0.0 ^ opprundifferential>=18.0 ^ oppdayssincegame<1.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured<0.0 ^ oppnuminjured>=0.0 ^ opprundifferential>=18.0 ^ oppdayssincegame>=1.0 ^ oppwinningpercent<0.0806987298889) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured<0.0 ^ oppnuminjured>=0.0 ^ opprundifferential>=18.0 ^ oppdayssincegame>=1.0 ^ oppwinningpercent>=0.0806987298889 ^ rundifferential<3.0 ^ winpercent>=0.699154368204) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured>=0.0 ^ oppwinningpercent<0.125358171942 ^ oppwinningpercent<0.0068583423205) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured>=0.0 ^ oppwinningpercent<0.125358171942 ^ oppwinningpercent>=0.0068583423205 ^ opprundifferential<23.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured>=0.0 ^ oppwinningpercent<0.125358171942 ^ oppwinningpercent>=0.0068583423205 ^ opprundifferential>=23.0 ^ winpercent<0.387374577582 ^ oppwinningpercent>=0.0651108904444) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured>=0.0 ^ oppwinningpercent>=0.125358171942 ^ oppwinningpercent<0.270328014607 ^ winpercent<0.289124011761) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured>=0.0 ^ oppwinningpercent>=0.125358171942 ^ oppwinningpercent<0.270328014607 ^ winpercent>=0.289124011761 ^ oppwinningpercent<0.189465166099 ^ winpercent<0.371677431261) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured>=0.0 ^ oppwinningpercent>=0.125358171942 ^ oppwinningpercent<0.270328014607 ^ winpercent>=0.289124011761 ^ oppwinningpercent>=0.189465166099 ^ opprundifferential<36.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured>=0.0 ^ oppwinningpercent>=0.125358171942 ^ oppwinningpercent<0.270328014607 ^ winpercent>=0.289124011761 ^

oppwinningpercent>=0.189465166099 ^ opprundifferential>=36.0 ^ winpercent<0.528333144456) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured>=0.0 ^ oppwinningpercent>=0.125358171942 ^ oppwinningpercent>=0.270328014607 ^ oppwinningpercent<0.307734761131 ^ winpercent<0.539602893417 ^ winpercent<0.539007439959 ^ oppdayssincegame<2.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured>=0.0 ^ oppwinningpercent>=0.125358171942 ^ oppwinningpercent>=0.270328014607 ^ oppwinningpercent<0.307734761131 ^ winpercent<0.539602893417 ^ winpercent<0.539007439959 ^ oppdayssincegame>=2.0)

3. (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured<0.0 ^ oppnuminjured<0.0 ^ winpercent>=0.590582828588 ^ oppwinningpercent>=0.494600828567 ^ winpercent>=0.600132770207)

“If the opponent has no injured players (also means less than 3 and less than 1 from the other splits) and we have no injured players (also means less than 1 from other split) and our win percentage is equal to or higher than 60.01% (also means higher than or equal to 59.06% from other split) and the opponent winning percentage is equal to or higher than 49.46%, then we are predicted to win the game”

4. We implemented top-down pruning. We first get the mode classification value for the validation set (whether winner is 1 or 0 the most). Then, we look at the accuracy of the current node. If the accuracy of just predicting the mode value for winner (blind guessing) is greater than the accuracy of the current node, we prune that entire branch and replace it with a leaf node with the mode winner value (0 or 1). As nodes higher up the tree are the ones with the best information gain ratio, no node below it will be better, and thus we can prune that entire branch. If the current node’s accuracy is higher, we recurse through its children.

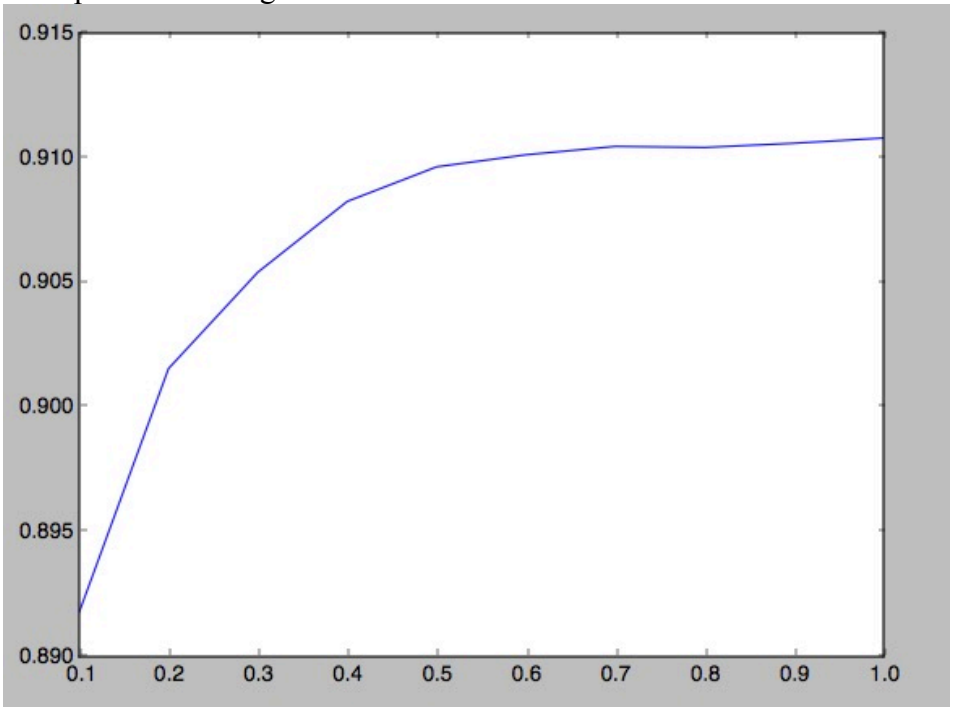
5. (restricted to 16 leaf nodes)
(oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured<0.0 ^ oppnuminjured<0.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured<0.0 ^ oppnuminjured>=0.0 ^ opprundifferential<30.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured>=0.0 ^ oppwinningpercent>=0.110564199499 ^ oppwinningpercent<0.277250882063 ^ winpercent<0.452110687309) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured<1.0 ^ numinjured>=0.0 ^ oppwinningpercent>=0.110564199499 ^ oppwinningpercent>=0.277250882063) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential<19.0 ^ rundifferential<12.0 ^ rundifferential<3.0 ^ weather=1) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential<19.0 ^ rundifferential<12.0 ^ rundifferential>=3.0 ^ opprundifferential<10.0 ^ winpercent<0.667310699054) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential<19.0 ^ rundifferential>=12.0 ^ rundifferential<30.0 ^ opprundifferential<8.0 ^ oppwinningpercent<0.379750315096 ^ winpercent<0.573437483861) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential<19.0

^ rundifferential>=12.0 ^ rundifferential<30.0 ^ opprundifferential<8.0 ^ oppwinningpercent>=0.379750315096) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential<19.0 ^ rundifferential>=12.0 ^ rundifferential<30.0 ^ opprundifferential>=8.0 ^ oppwinningpercent>=0.527969069748 ^ rundifferential>=22.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential<19.0 ^ rundifferential>=12.0 ^ rundifferential>=30.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential>=19.0 ^ opprundifferential<23.0 ^ rundifferential>=71.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential>=19.0 ^ opprundifferential>=23.0 ^ opprundifferential<43.0 ^ rundifferential<53.0 ^ winpercent>=0.409365261398 ^ temperature<74.4658041955 ^ rundifferential<14.0 ^ rundifferential<2.0) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential>=19.0 ^ opprundifferential>=23.0 ^ opprundifferential<43.0 ^ rundifferential>=53.0 ^ rundifferential<104.0 ^ rundifferential<87.0 ^ opprundifferential<27.0 ^ oppwinningpercent>=0.257787028218 ^ oppwinningpercent>=0.324607506889 ^ rundifferential>=54.0 ^ startingpitcher=1) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential>=19.0 ^ opprundifferential>=23.0 ^ opprundifferential<43.0 ^ rundifferential>=53.0 ^ rundifferential<104.0 ^ rundifferential<87.0 ^ opprundifferential<27.0 ^ oppwinningpercent>=0.257787028218 ^ oppwinningpercent>=0.324607506889 ^ rundifferential>=54.0 ^ startingpitcher=2) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential>=19.0 ^ opprundifferential>=23.0 ^ opprundifferential<43.0 ^ rundifferential>=53.0 ^ rundifferential<104.0 ^ rundifferential<87.0 ^ opprundifferential<27.0 ^ oppwinningpercent>=0.257787028218 ^ oppwinningpercent>=0.324607506889 ^ rundifferential>=54.0 ^ startingpitcher=3) v (oppnuminjured<3.0 ^ numinjured<1.0 ^ oppnuminjured>=1.0 ^ opprundifferential>=19.0 ^ opprundifferential>=23.0 ^ opprundifferential<43.0 ^ rundifferential>=53.0 ^ rundifferential<104.0 ^ rundifferential<87.0 ^ opprundifferential<27.0 ^ oppwinningpercent>=0.257787028218 ^ oppwinningpercent>=0.324607506889 ^ rundifferential>=54.0 ^ startingpitcher=4)

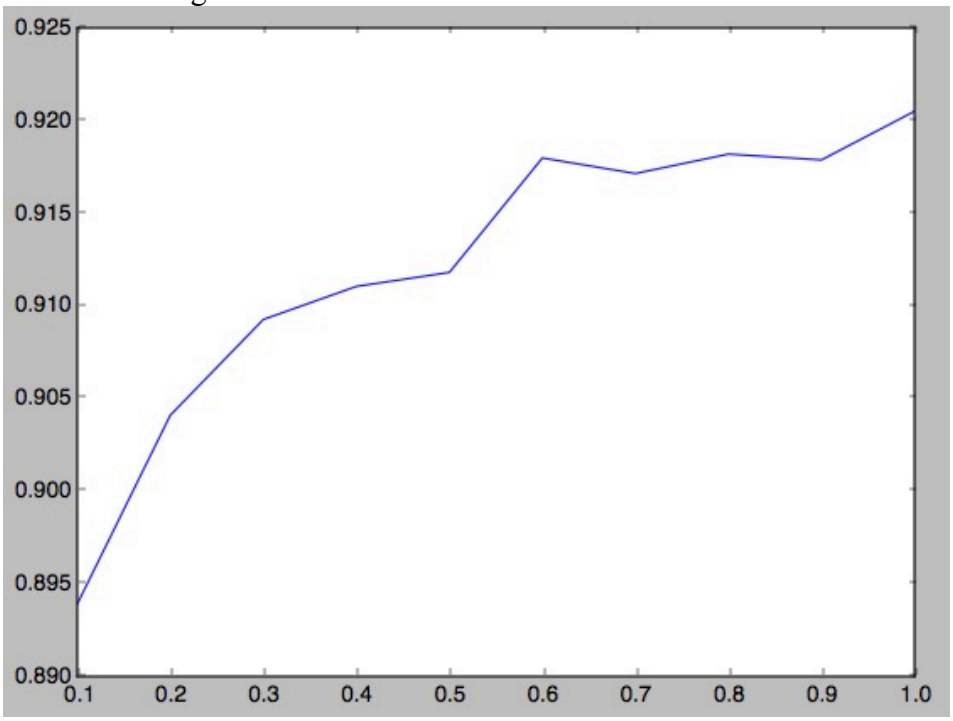
6. Splits for the **unpruned** tree: 2,799 Splits for the **pruned** tree: 439
Split difference: 2,360

7. **Unpruned** accuracy: `0.916176470588` **Pruned** accuracy: `0.924474789916`
There is a slight difference in validation accuracy between the two, with the pruned tree being more accurate. This is because the depth and numeric split count values allowed for overfitting in the unpruned tree, and thus caused a decrease in validation accuracy. Pruning the tree helped to eliminate the overfitting. However, the pruned accuracy isn't much higher than the unpruned accuracy. This is because we didn't allow the unpruned tree to way overfit the training data due to time constraints (by increasing depth or split counts).

8. Unpruned learning curve:



Pruned learning curve:



The pruned learning curve seems to have higher accuracy at every point as compared to the unpruned learning curve. It seems that amount of information doesn't change the fact that our pruned tree has higher validation accuracy than the unpruned tree.

9. We think the pruned tree will perform better on the unlabeled test set. Pruning helps deal with overfitting, which makes it more applicable to an unseen example, as shown with the validation accuracy difference. Whatever has the higher validation accuracy should do better on the test data. This is backed up by the graph on slide 43 of the Decision Trees lecture. The graph shows pruned trees consistently performing better on test data than unpruned trees.

10. Most of the work done for the assignment was worked on together with pair programming. However, we can try to split it up based on who did the majority of each part.
I (Jonathan) did most of:
ID3 functions

Alex did most of:
Node functions
Pruning functions
Graphing functions

11. First, using IGR instead of using just IG was a good model decision. With pruning, the validation accuracy using IGR was `0.924474789916`. Changing this to using just IG (still with pruning) actually dropped the validation accuracy to `0.913865546218`. This difference in accuracy is bigger than the accuracy difference between pruning and not pruning. Additionally, IGR isn't much more computationally expensive, meaning it is clearly the better option for the model. However, changing step size to include every possible threshold value in the data set is computationally prohibitive. While it may very well increase accuracy, the large difference in training time means using larger step sizes might be the better choice for the model.