DD2421 Machine Learning - Lab 1: Decision Trees

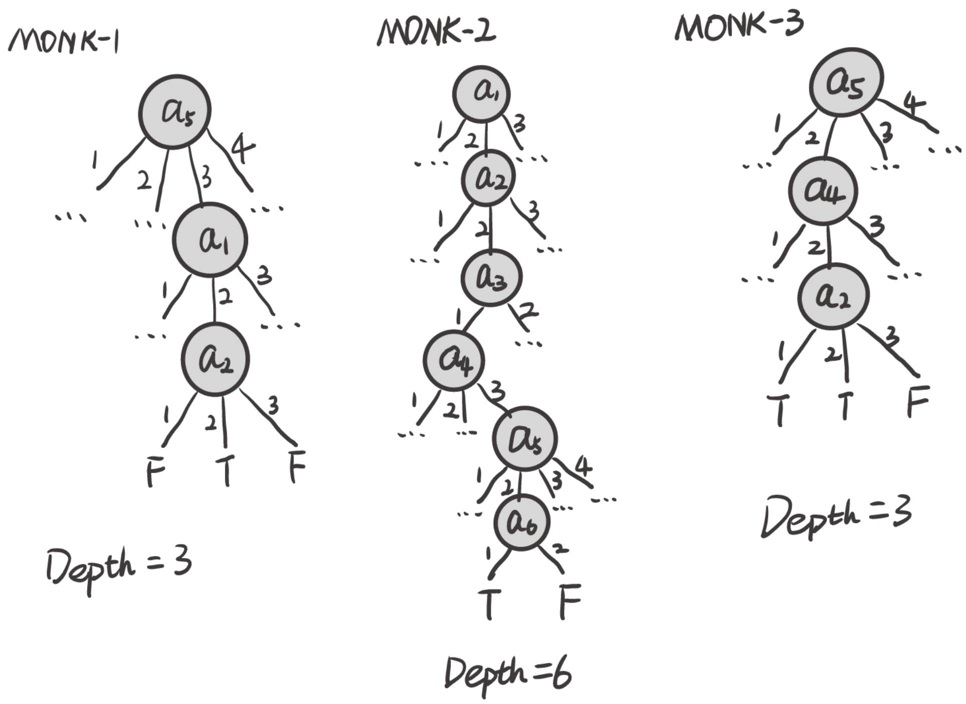
Chen Yuan, Yuxuan Zhang

Assignment 0:

Each one of the datasets has properties which makes them hard to learn. Motivate which of the three problems is most difficult for a decision tree algorithm to learn.

MONK-2 is most difficult to learn.

The simplest trees with the worst cases:



Other properties that add difficulties to learning:

MONK-1 and MONK-3 have smaller training set than MONK-2; MONK-3 has 5% additional noise (misclassification) in the training set.

Assignment 1:

The file dtree.py defines a function entropy which calculates the entropy of a dataset. Import this file along with the monks datasets and use it to calculate the entropy of the training datasets.

```
Entropy of MONK-1:
1.0
Entropy of MONK-2:
0.957117428264771
Entropy of MONK-3:
0.9998061328047111
```

Assignment 2:

Explain entropy for a uniform distribution and a non-uniform distribution, present some example distributions with high and low entropy.

Uniform distribution:

Example: tossing a coin
$p_{\text{head}} = 0.5; \qquad p_{\text{tail}} = 0.5$

$$\text{Entropy} = \sum_i -p_i \log_2 p_i =$$
$$= -0.5 \underbrace{\log_2 0.5}_{-1} - 0.5 \underbrace{\log_2 0.5}_{-1} =$$
$$= 1$$

Non-uniform distibution:

$$p_{head} = 0.1;\ p_{tail} = 0.9$$
$$Entropy = \sum_i -p_i log_2 p_i = -0.1 log_2 0.1 - 0.9 log_2 0.9 \approx 0.47$$

Entropy for a uniform distribution is higher than that for a non-uniform distribuiton, which means the uniform distribution is harder to predict.

Assignment 3:

Use the function averageGain (defined in dtree.py) to calculate the expected information gain corresponding to each of the six attributes. Note that the attributes are represented as instances of the class Attribute (defined in monkdata.py) which you can access via m.attributes[0], ..., m.attributes[5]. Based on the results, which attribute should be used for splitting the examples at the root node?

```
MONK-1:
a1 0.07527255560831925
a2 0.005838429962909286
a3 0.00470756661729721
a4 0.02631169650768228
a5 0.28703074971578435
a6 0.0007578557158638421
Therefore, choose a5 for MONK-1.
```

```
MONK-2:
a1 0.0037561773775118823
a2 0.0024584986660830532
a3 0.0010561477158920196
a4 0.015664247292643818
a5 0.01727717693791797
a6 0.006247622236881467
Therefore, choose a5 for MONK-2.
```

```
MONK-3:
a1 0.007120868396071844
a2 0.29373617350838865
a3 0.0008311140445336207
a4 0.002891817288654397
a5 0.25591172461972755
a6 0.007077026074097326
Therefore, choose a2 for MONK-3.
```

Assignment 4:

For splitting we choose the attribute that maximizes the information gain, Eq.3. Looking at Eq.3 how does the entropy of the subsets, Sk, look like when the information gain is maximized? How can we motivate using the information gain as a heuristic for picking an attribute for splitting? Think about reduction in entropy after the split and what the entropy implies.

$$\mathrm{Gain}(S, A) = \mathrm{Entropy}(S) - \sum_{k \in \mathrm{values}(A)} \frac{|S_k|}{|S|} \mathrm{Entropy}(S_k) \qquad (3)$$

The entropy of S_k should be as small as possible to maximize the information gain.

When choosing certain attribute A of the total set S as the splitter, S is then splitted into serveral subsets with different values of attribute A (S_k). As the entropy is the measurement of uncertainty, the smaller entropy of S_k indicates that the splitted subsets are easier to be predicted due to lower uncertainty. And at the same time, the information gain becomes larger. Therefore, when maximizing the information gain, we are ensuring that the selected attribute is splitting the total set into subsets that are the easiest to be predicted.

Assignment 5:

Split the monk1 data into subsets according to the selected attribute using the function select (again, defined in dtree.py) and compute the information gains for the nodes on the next level of the tree. Which attributes should be tested for these nodes?

The selected attribute is a5 (as the root node). The subsets relate to $a_5 \in \{1, 2, 3, 4\}$. The information gains for the four subsets w.r.t. different attributes are:

```
For the subset with a5 = 1
a0: 0.0
a1: 0.0
a2: 0.0
a3: 0.0
a4: 0.0
a5: 0.0
```

```
For the subset with a5 = 2
a0: 0.040216841609413634
a1: 0.015063475072186083
a2: 0.03727262736015946
a3: 0.04889220262952931
a4: 0.0
a5: 0.025807284723902146
```

```
For the subset with a5 = 3
a0: 0.03305510013455182
a1: 0.002197183539100922
a2: 0.017982293842278896
a3: 0.01912275517747053
a4: 0.0
a5: 0.04510853782483648
```

```
For the subset with a5 = 4
a0: 0.20629074641530198
a1: 0.033898395077640586
a2: 0.025906145434984817
a3: 0.07593290844153944
a4: 0.0
a5: 0.0033239629631565126
```

Therefore, for the subset with $a_5 = 1$, all information gains are 0 as the tree comes to an end after $a_5 = 1$ for MONK-1, thus no new information is gained.

For the subset with $a_5 = 2$, select a_4.

For the subset with $a_5 = 3$, select a_6.

For the subset with $a_5 = 4$, select a_1.

For the monk1 data draw the decision tree up to the first two levels and assign the majority class of the subsets that resulted from the two splits to the leaf nodes. You can use the predefined function mostCommon (in dtree.py) to obtain the majority class for a dataset.

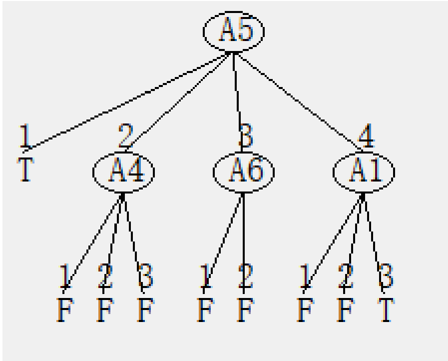
```
For a5 = 1:
True

For a5 = 2:
a4 = 1:
False
a4 = 2:
False
a4 = 3:
False
```

```
For a5 = 3:
a6 = 1:
False
a6 = 2:
False

For a5 = 4:
a1 = 1:
False
a1 = 2:
False
a1 = 3:
True
```

Now compare your results with that of a predefined routine for ID3. Use the function buildTree(data, m.attributes) to build the decision tree. If you pass a third, optional, parameter to buildTree, you can limit the depth of the generated tree.



Build the full decision trees for all three Monk datasets using buildTree. Then, use the function check to measure the performance of the decision tree on both the training and test datasets.

For example to built a tree for monk1 and compute the performance on the test data you could use

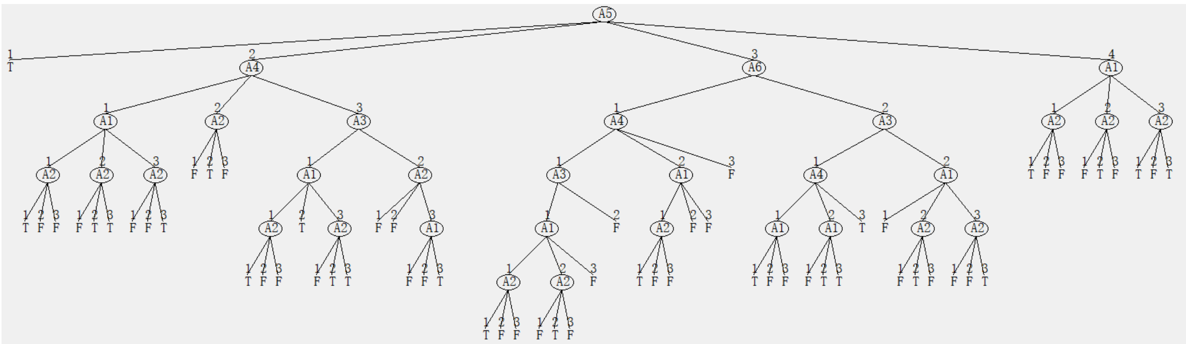
import monkdata as m

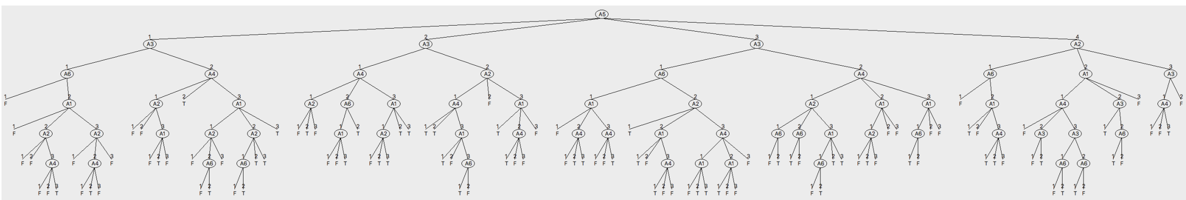
import dtree as d

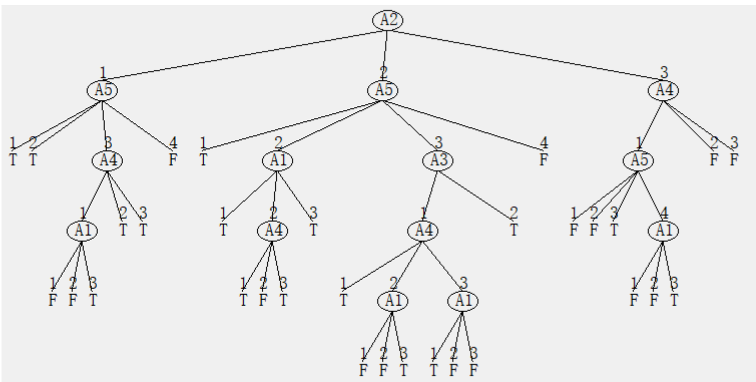
t=d.buildTree(m.monk1, m.attributes);

print(d.check(t, m.monk1test))

Compute the train and test set errors for the three Monk datasets for the full trees. Were your assumptions about the datasets correct? Explain the results you get for the training and test datasets.







```
Etrain:
MONK-1: 0.0 MONK-2: 0.0 MONK-3: 0.0
Etest:
MONK-1: 0.17129629629629628 MONK-2: 0.30787037037037035 MONK-3: 0.05555555555555558
```

The result is in line with our assumption that MONK-2 has the largest test error, as we assumed that MONK-2 is the hardest to learn for the complexity of its decision tree.

The training sets errors are all 0 because the trees are constructed with the training set data.

The errors may be due to overfitting. As the images of the trees show, the more complex trees result in the higher error. Besides, as simplifying the model and accepting some errors for the training examples may alleviate overfitting, MONK-3 get a lower error probably because its training set has some noise.

Assignment 6

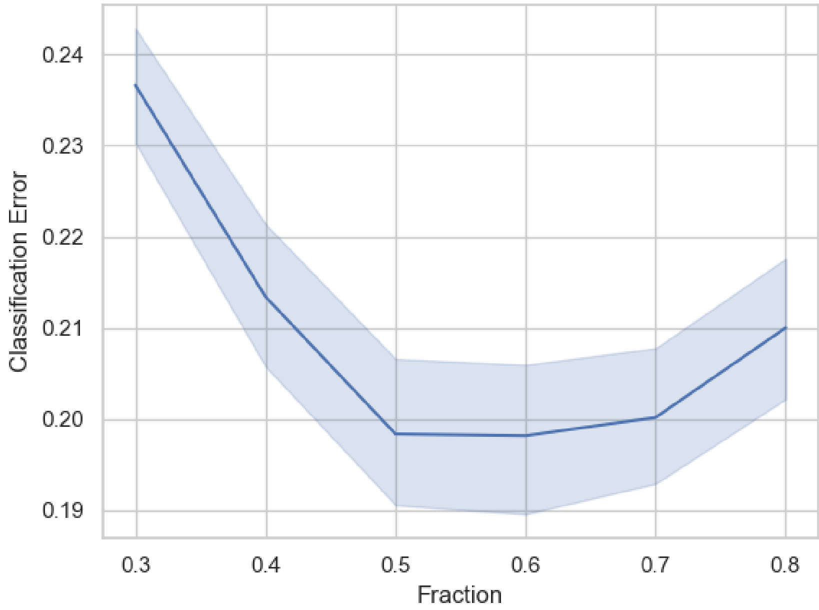
Explain pruning from a bias variance trade-off perspective.

Pruning a decision tree is simplifying it, thus reducing the variance while increasing the bias. With the control of the pruning extent, the variance can be largely reduced with a small increase in bias, thus increasing the accuracy of the model.

Assignment 7

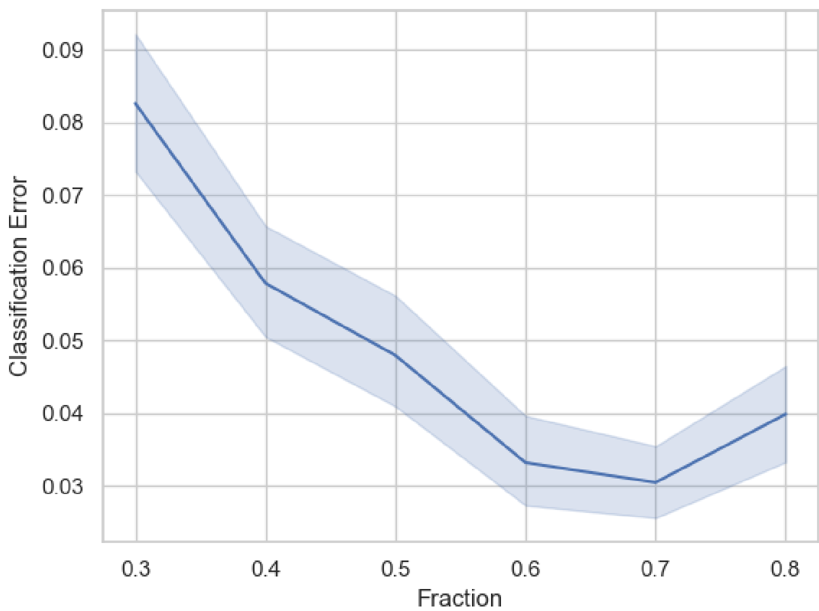
With 100 runs, MONK-1:

```
Mean:
          Classification Error
Fraction
0.3                   0.236620
0.4                   0.213426
0.5                   0.198356
0.6                   0.198171
0.7                   0.200162
0.8                   0.210000
Std:
          Classification Error
Fraction
0.3                   0.033119
0.4                   0.041021
0.5                   0.040388
0.6                   0.040410
0.7                   0.037674
0.8                   0.040912
```



MONK-3:

```
Mean:
          Classification Error
Fraction
0.3                   0.082593
0.4                   0.057870
0.5                   0.047940
0.6                   0.033194
0.7                   0.030463
0.8                   0.039861
Std:
          Classification Error
Fraction
0.3                   0.047577
0.4                   0.037476
0.5                   0.038839
0.6                   0.031255
0.7                   0.025368
0.8                   0.032782
```



The smallest errors exist somewhere between 0.5 and 0.7 as larger training sets result in higher accuracy. There is a small increase when $fraction = 0.8$. This may be because the validation set is so small that the pruning function leads to more errors.