TDT4171: Methods in AI

Assignment 4

Anders Kjelsrud

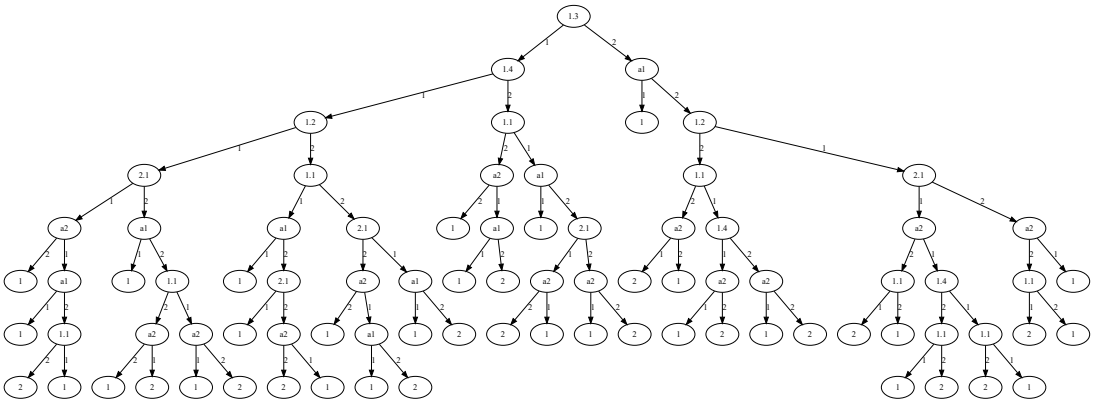
March 2022

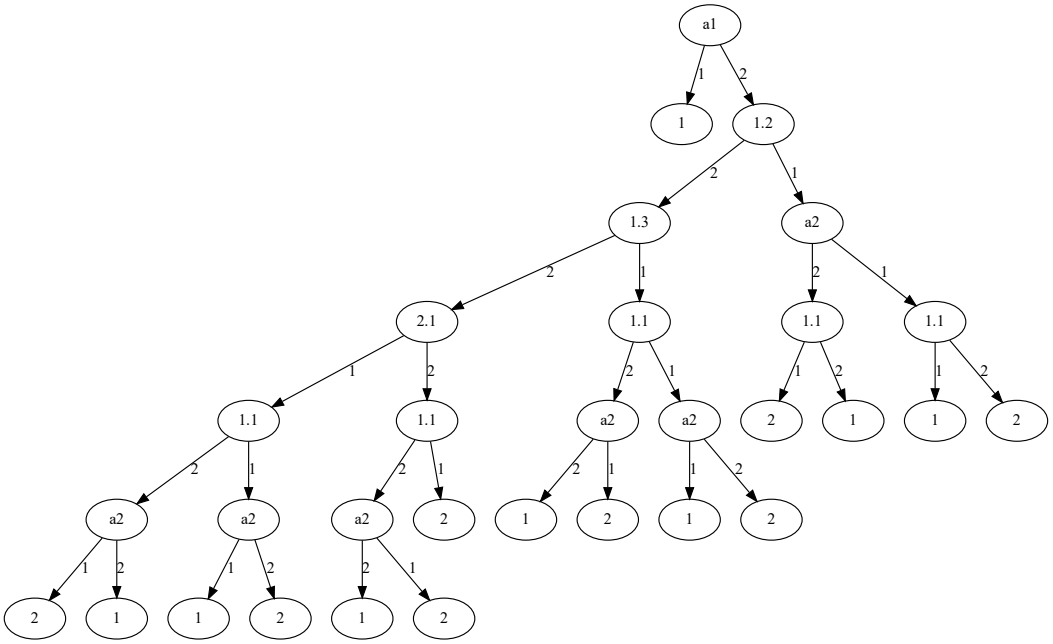
1 Implementation

Calling the function

```
learn_decision_tree()
```

with the flag `rand = True` uses the implementation allocating random values as importance for the attributes produced the following tree. To clarify I renamed the attributes '1' and '2' to 'a1' and 'a2' for readability. The implementation using information gain as importance constructed a





more shallow tree as shown above. Running the script `decisiontree.py` will print the accuracy of the model tested on the test-data. The accuracy for the model using information gain is

```
Model predicted 26 correct and 1 wrong.
Accuracy = 0.963
```

while the accuracy for the model using random values as importance is

```
Model predicted 27 correct and 0 wrong.
Accuracy = 1.0
```

This is somewhat surprising, but looking at the trees created, the one from random importance is a lot more complex and could be the reason it scores perfectly. In fact the model using information gain will classify about half of the cases by only doing 1 comparison, because approximately half of the test-set has `a1 = 1`.

2 Discussion

- The resulting accuracy on the test-set is very similar, with the model using information gain coming up short with 1 mistake. The one using random importance got them all correct. The success did however come at a cost, which is a substantially more complicated tree requiring many more comparisons to correctly classify training data. Therefore I will conclude that the model using information gain as importance is the best one.
- Running the learner with random importance given to the attributes several times produces different learning trees. This is expected because the algorithm chooses what attribute to split on based on the highest importance and will thus choose different splits based on the random values.
- Naturally, when running the learner based on information gain several times, the same model is learned. Because the same importance is given to the corresponding attribute every time, the attributes it splits on will be the same, and thus the model will also remain the same.