Charles Hancock

Question 5:

The tree generated for the Connect 4 dataset has size 41521 based on 67557 examples and accuracy ~75% whereas the tree for the cars dataset has size 408 based on 1728 examples and accuracy just south of 90%. The cars tree is much smaller compared to input size than the connect 4 tree; it is also more accurate. I believe this is because there are attributes of the cars dataset that correlate extremely highly with the classifications. Indeed, a peek at car.out reveals that any low safety car is guaranteed to be unacceptable, meaning that cutting the tree at the very top level on the safety attribute gets rid of a very large amount of examples, unless there are very very few low safety cars. The connect 4 dataset has much more randomness and no one attribute is correlated with classification highly enough to result in as much efficiency or accuracy.

The dummy sets had sizes of 3 and 11 and accuracies of 100% and 55%. I suppose they conform to the observations I make about cars and connect4 before, as they both have input size of 20, and the first is way more efficient. Looking through dataInterface.py confirms my suspicions, as the index 5 entries in the first dataset have a perfect correlation with the labels whereas the second does not exhibit this behavior.

Question 6:

The decision tree works just like the cars.com commercials (the ones where they say How about an SUV? Now only the ones with 5-star safety. Now the black ones. Now under $10,000, etc.) The customer pares down the massive example set using his or her most important dimensions until they arrive at a single perfect car. An agent that learns this criteria for different people would be very useful in a retail environment, but probably for one that facilitated repeat purchases. I'm sure Amazon.com already uses things like this in their suggestion algorithms.

The Connect 4 game analysis could help an agent figure out heuristics for where the "power spots" are on the board. It could even, with enough datapoints, start finding power spots against certain opponents. This would be most effective in Minimaxing/Expectimaxing when you need to evaluate a game state.