Question 6:

```
C:\Users\Alan\PycharmProjects\Project4a\Project4a>mine.py
td1 Testing dummy dataset 1. Number of examples 20.
Tree is as follows:
5 = 0
|---1
5 = 1
|---0


Tree size: 3.

Results for training set:
(1.0, [])

Done
```

```
(<DecisionTree.Tree instance at 0x02937878>, (1.0, []))
td2 Testing dummy dataset 2. Number of examples 20.
Tree is as follows:
2 = 0
|---0 = 0
|---|---0
|---0 = 1
|---|---4 = 0
|---|---|---1
|---|---4 = 1
|---|---|---0
2 = 1
|---5 = 0
|---|---6 = 0
|---|---|---0
|---|---6 = 1
|---|---|---1
|---5 = 1
|---|---1
```

```
Tree size: 11.

Results for training set:
(0.65, [(0, 1), (0, 1), (1, 0), (0, 1), (0, 1), (0, 1), (1, 0)])

Done
```
____

```
(<DecisionTree.Tree instance at 0x02937E68>, (0.65, [(0, 1), (0, 1), (1, 0), (0, 1), (0, 1), (0, 1), (1,
0)]))
Connect4 Testing Connect4 dataset. Number of examples 67557.
Tree size: 41521.

Entire tree written out to connect4.out in local directory

Starting test for average error for 10 runs with test size 2000
Score for run 1 is 0.740000
Score for run 2 is 0.779000
Score for run 3 is 0.756000
Score for run 4 is 0.770000
Score for run 5 is 0.759000
Score for run 6 is 0.757000
Score for run 7 is 0.752000
Score for run 8 is 0.753000
Score for run 9 is 0.756500
Score for run 10 is 0.768500
Average classification rate over all runs: 0.759100
Results for training set:
([0.74, 0.779, 0.756, 0.77, 0.759, 0.757, 0.752, 0.753, 0.7565, 0.7685], 0.7590999999999999)

Done
```
____

```
(<DecisionTree.Tree instance at 0x0E3E4260>, ([0.74, 0.779, 0.756, 0.77, 0.759, 0.757, 0.752,
0.753, 0.7565, 0.7685], 0.7590999999999999))
testCar Testing Car dataset. Number of examples 1728.
Tree size: 408.
```

```
Entire tree written out to car.out in local directory

Starting test for average error for 20 runs with test size 200
Score for run 1 is 0.955000
Score for run 2 is 0.945000
Score for run 3 is 0.950000
Score for run 4 is 0.925000
Score for run 5 is 0.920000
Score for run 6 is 0.950000
Score for run 7 is 0.955000
Score for run 8 is 0.955000
Score for run 9 is 0.950000
Score for run 10 is 0.955000
Score for run 11 is 0.925000
Score for run 12 is 0.925000
Score for run 13 is 0.940000
Score for run 14 is 0.935000
Score for run 15 is 0.930000
Score for run 16 is 0.940000
Score for run 17 is 0.920000
Score for run 18 is 0.935000
Score for run 19 is 0.975000
Score for run 20 is 0.935000
Average classification rate over all runs: 0.941000
Results for training set:
([0.955, 0.945, 0.95, 0.925, 0.92, 0.95, 0.955, 0.955, 0.95, 0.955, 0.925, 0.925, 0.94, 0.935, 0.93,
0.94, 0.92, 0.935, 0.975, 0.935], 0.9410000000000001)

Done
```

```
(<DecisionTree.Tree instance at 0x0E3E4210>, ([0.955, 0.945, 0.95, 0.925, 0.92, 0.95, 0.955,
0.955, 0.95, 0.955, 0.925, 0.925, 0.94, 0.935, 0.93, 0.94, 0.92, 0.935, 0.975, 0.935],
0.9410000000000001))
```

For testDummySet1, the tree size is tiny (3) and the classification rate is 1, because the number of examples is very small (20) and the decision tree is extremely simple so there is no risk of overfitting.
For testDummySet2, the tree size is small (11) because the example number is low (20) but the classification rate is only ~0.65 because the decision tree is too complex relative to the number of examples, meaning it is overfitting the training data.
For Connect4, the tree size is large (~41k) relative to the large number of examples (~67k) and the decision tree is overfitting the training data. The lack of pruning worsens this, which explains why the classification rate is limited to ~0.75. Because the decision tree takes into account all the variations on already-decided scenarios, in which black or red has already won (gotten 4 in a row) but the other pieces are arranged differently, this exacerbates the overfitting problem.
For Car, the tree size is smaller (~400) relative to the number of examples (~1700) and because we have fewer attributes with which to construct the decision tree, overfitting is less likely. The classification rate is higher, at ~0.95, because unlike Connect4, this decision tree does not worry about the irrelevant attribute values after the terminal conditions are met.

Question 7:

The cars dataset and decision tree could be applied analogously to the idea of selling homes. A company such as Redfin, which applies big data to real estate, could use a decision tree to determine the likelihood/importance of various factors in deciding whether or not to buy the home. Price (greater or less than $X00,0000), #bedrooms (greater or less than Z), basement (Y/N), garage (Y/N), location (urban/rural), etc, are all potential attributes for the decision tree. More significant variables would be higher in the tree, so for instance price would likely be near the top because it is usually the most important consideration for buying a home. These attributes would be advertised more aggressively, or otherwise emphasized in the sales strategy. The attributes could be adjusted to whatever data is provided by MLS listings.

In the case of Connect4, we could use the decision tree to form a heuristic for a graph search algorithm, like bestFS or A*. Since the decision tree places nodes corresponding to moves that give more information (e.g. place a red piece at E7 and probability of winning goes to 0.21) higher in the decision tree, we can assign the nodes corresponding to these moves larger values for better/worse in the graph heuristic as appropriate. By associating “good” moves with higher values in the heuristic and “bad” moves with lower values (since negative heuristic values might break A*) we can use the search algorithms to help us win at Connect 4.