Sarah Kunzler

CS472

10/29/19

Decision Tree Lab Report

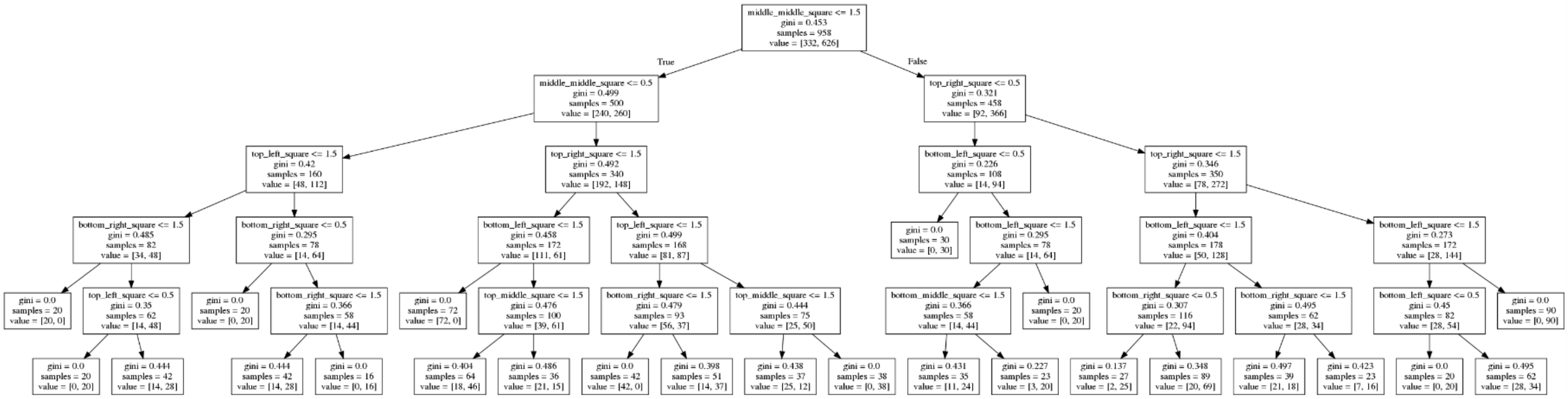
1. Accuracy = [0.15]. CSV attached.
2. Cars and Voting data

Cars	.8092	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.6279
	Avg: 0.9437									
Voting	0.9545	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9070
	Avg: 0.9862									

   I got 100% accuracy the majority of the time, which tells me that the training data and the test data generally followed the same paths down the tree, and that each particular set of data led to a particular output most of the time. In the cases where my accuracy was not 100% I assume that my training data had sets of values not represented in my testing data, so it had to guess based on what testing data I had.
3. Voting: My tree splits first on
   a. For the cars dataset, my tree splits on safety first. This makes sense, as most people are extremely concerned with buying a vehicle they will be safe in. After that is price and persons, which says that people also care about how expensive the car is and how many people it will hold. The next level of the tree splits on maintenance expenses, price again, and the size of the luggage boot, also important factors in choosing a car.
   b. For the voting dataset my tree splits first on the physician fee freeze issue. This tells us that an individual's opinion on this issue is highly correlated to how they will vote. After that the tree split on adoption of the budget resolution, synfuels corporation cutback, and the mx missile. The next level split on education spending, handicapped infants, export administration act south Africa, and water project cost sharing. These could tell political parties what issues to play up if they want to gain voters, since they are apparently such decisive issues.
4. I chose to make unknown data its own variable. I did this because often when people leave an answer blank it can tell us something about their choice, especially in voting. For example, perhaps one of the parties appeal more to people without strong opinions on a matter. If so, that is something we would want to account for.
5. Sklearn models
   a. Voting Avg Accuracies:
      i. No params: 0.963
      ii. Max_depth=6: 0.965
      iii. max_leaf_nodes=35: 0.967
      iv. min_samples_split=8: 0.9685
         1. *Note: I was adding each of these on. So the avg accuracy for max_depth was with only that parameter, but the accuracy for min_samples had all 3.

   b. Cars Avg Accuracies
      i. No params: 0.8848
      ii. 3 params like above: 0.807
      iii. Everything I try seems to give me a lower score. So I would choose to run this with no params
   c. Mine gets about the same accuracy on the Voting problem, but actually does better on the Cars. That actually makes me a bit nervous about my algorithm.

6. Tic Tac Toe Tree: I had to play with the hyperparameters a bit to make this tree small enough to fit on a page which actually brought my accuracy down a bit. However, the tree shows us that the most important square you can take to determine a game is the middle square. After that is the top right square. On the next level we see the next most significant moves are the top and bottom left squares. These make sense because the middle and corner squares present the most possible moves and ways to win.


middle_middle_square <= 1.5 gini = 0.453 samples = 958 value = [332, 626]
True
False
middle_middle_square <= 0.5 gini = 0.499 samples = 500 value = [240, 260]
top_right_square <= 0.5 gini = 0.321 samples = 458 value = [92, 366]
top_left_square <= 1.5 gini = 0.42 samples = 160 value = [48, 112]
top_right_square <= 1.5 gini = 0.492 samples = 340 value = [192, 148]
bottom_left_square <= 0.5 gini = 0.226 samples = 108 value = [14, 94]
top_right_square <= 1.5 gini = 0.346 samples = 350 value = [78, 272]
bottom_right_square <= 1.5 gini = 0.485 samples = 82 value = [34, 48]
bottom_right_square <= 0.5 gini = 0.295 samples = 78 value = [14, 64]
bottom_left_square <= 1.5 gini = 0.458 samples = 172 value = [111, 61]
top_left_square <= 1.5 gini = 0.499 samples = 168 value = [81, 87]
gini = 0.0 samples = 30 value = [0, 30]
bottom_left_square <= 1.5 gini = 0.295 samples = 78 value = [14, 64]
bottom_left_square <= 1.5 gini = 0.404 samples = 178 value = [50, 128]
bottom_left_square <= 1.5 gini = 0.273 samples = 172 value = [28, 144]
gini = 0.0 samples = 20 value = [20, 0]
top_left_square <= 0.5 gini = 0.35 samples = 62 value = [14, 48]
gini = 0.0 samples = 20 value = [0, 20]
bottom_right_square <= 1.5 gini = 0.366 samples = 58 value = [14, 44]
gini = 0.0 samples = 72 value = [72, 0]
top_middle_square <= 1.5 gini = 0.476 samples = 100 value = [39, 61]
bottom_right_square <= 1.5 gini = 0.479 samples = 93 value = [56, 37]
top_middle_square <= 1.5 gini = 0.444 samples = 75 value = [25, 50]
bottom_middle_square <= 1.5 gini = 0.366 samples = 58 value = [14, 44]
gini = 0.0 samples = 20 value = [0, 20]
bottom_right_square <= 0.5 gini = 0.307 samples = 116 value = [22, 94]
bottom_right_square <= 1.5 gini = 0.495 samples = 62 value = [28, 34]
bottom_left_square <= 0.5 gini = 0.45 samples = 82 value = [28, 54]
gini = 0.0 samples = 90 value = [0, 90]
gini = 0.0 samples = 20 value = [0, 20]
gini = 0.444 samples = 42 value = [14, 28]
gini = 0.444 samples = 42 value = [14, 28]
gini = 0.0 samples = 16 value = [0, 16]
gini = 0.404 samples = 64 value = [18, 46]
gini = 0.486 samples = 36 value = [21, 15]
gini = 0.0 samples = 42 value = [42, 0]
gini = 0.398 samples = 51 value = [14, 37]
gini = 0.438 samples = 37 value = [25, 12]
gini = 0.0 samples = 38 value = [0, 38]
gini = 0.431 samples = 35 value = [11, 24]
gini = 0.227 samples = 23 value = [3, 20]
gini = 0.137 samples = 27 value = [2, 25]
gini = 0.348 samples = 89 value = [20, 69]
gini = 0.497 samples = 39 value = [21, 18]
gini = 0.423 samples = 23 value = [7, 16]
gini = 0.0 samples = 20 value = [0, 20]
gini = 0.495 samples = 62 value = [28, 34]