Assignment 1: Decision Tree

20.12.2021

Sevde SARIKAYA
2017400081

Description

In this project, we are expected to implement a decision tree algorithm with decision boundaries using entropy calculations.

Algorithm:

My algorithm can work for different depth values. It works recursively.

calEntropy: It is used to calculate entropy of an area. If you want to calculate left entropy of the boundary, it finds the points on the left side and calculates the entropy by summing all -p_i*log(p_i) values

findIG: It is used to calculate information gain. It calculates data entropy, left and right entropies of the boundary and returns these values.

$$IG = Entropy(Data) - \sum_{i=left,right} \frac{|S_i|}{|Data|} Entropy(S_i)$$

$Data = S_{left} \cup S_{right}$
$|S_{i=right}|$ = # samples in region i=right
$|Data|$ = # samples in the dataset

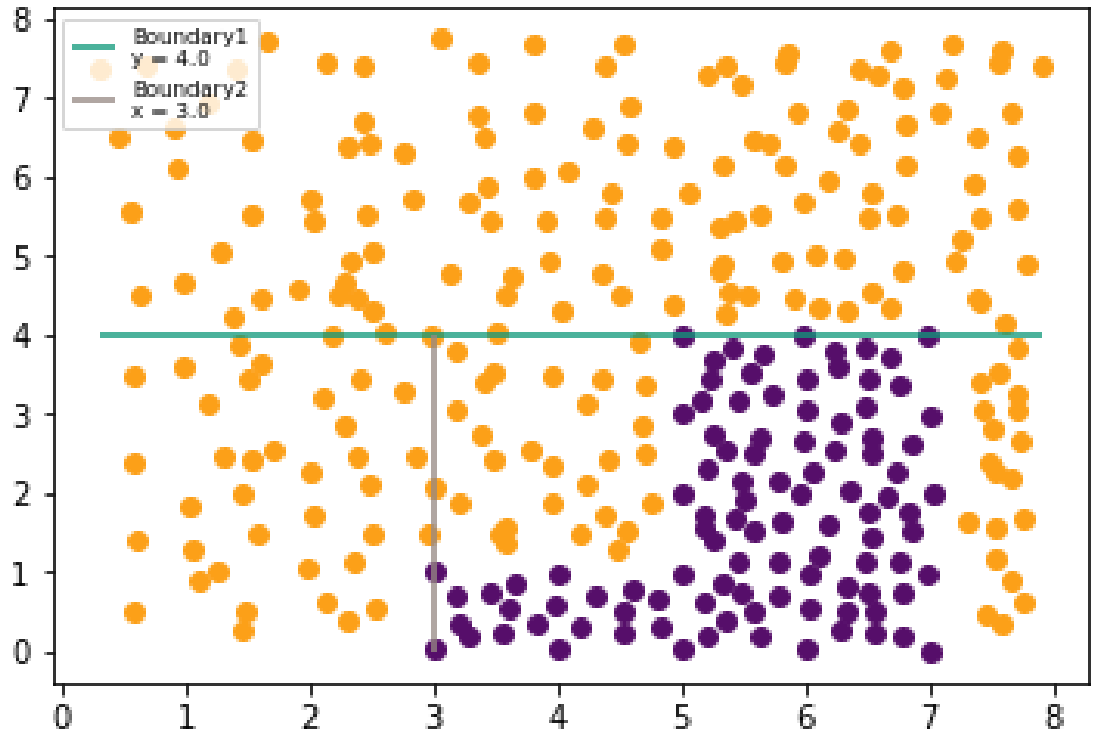
findBest: It tries different boundary values (for each x and y) until the last point in the data. Returns the information related to the boundary which gives the maximum information gain.

getMaxMin: It is used to get maximum and minimum of x and y values.

split: It runs recursively and stops when the tree reaches the maximum depth. It calls findBest in every iteration to find the boundary. Then, if the entropy is not zero, it calls itself again with the data in the left side of the boundary and the right side of the boundary. So, it calls itself twice in each iteration.

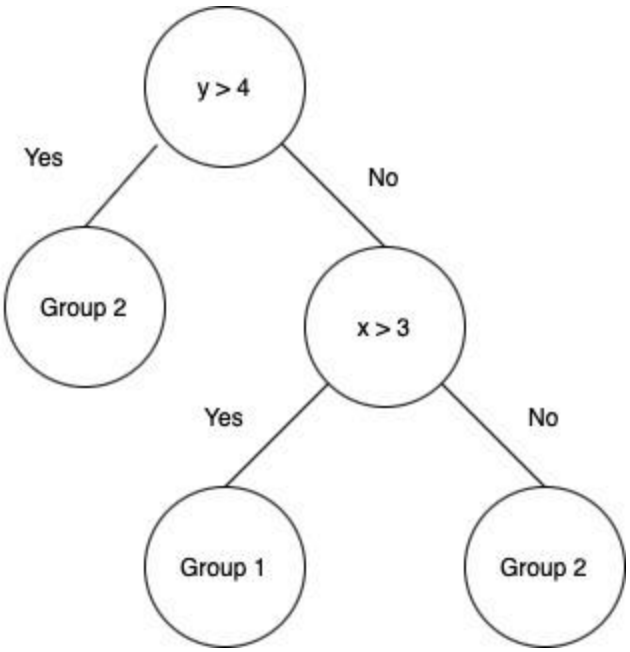
execute: It calls the split function with the given data and depth values. It prints and plots necessary information.

1.Data & Boundaries:



- Level 0 Entropy: 0.9117517586347538
- Entropy of the left of level 0: 0.9940302114769565
- Entropy of the right of level 0: 0.0
- Weighted Entropy of level 0: 0.595780928032663
- Entropy of the left of level 1: 0.0
- Entropy of the right of level 1: 0.899349319724299
- Weighted Entropy of level 1: 0.71659384298888

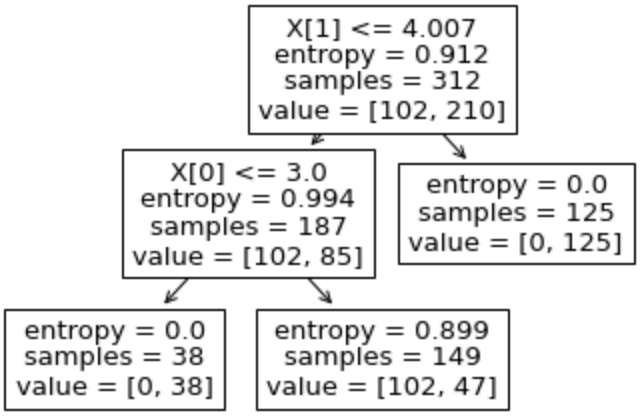
2.Plot of the Decision Tree:



Group 2: Orange Dots (Butterflies)

Group 1: Purple Dots(Birds)

3.Comparison with scikit-learn:



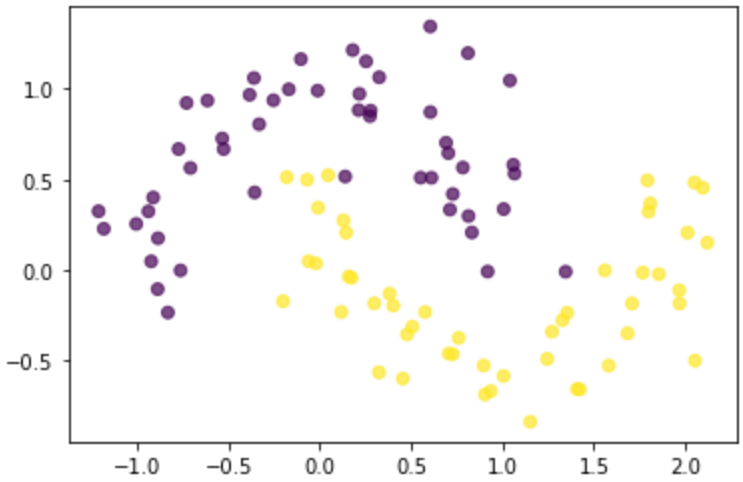
- Boundary 1: Y = 4.007,
- Boundary 2: X = 3.0

As one can see, my algorithm produces the same boundaries and entropy values with the scikit-learn. However, there is a precision loss (0.007) because my algorithm checks for boundaries with 0.01 step.

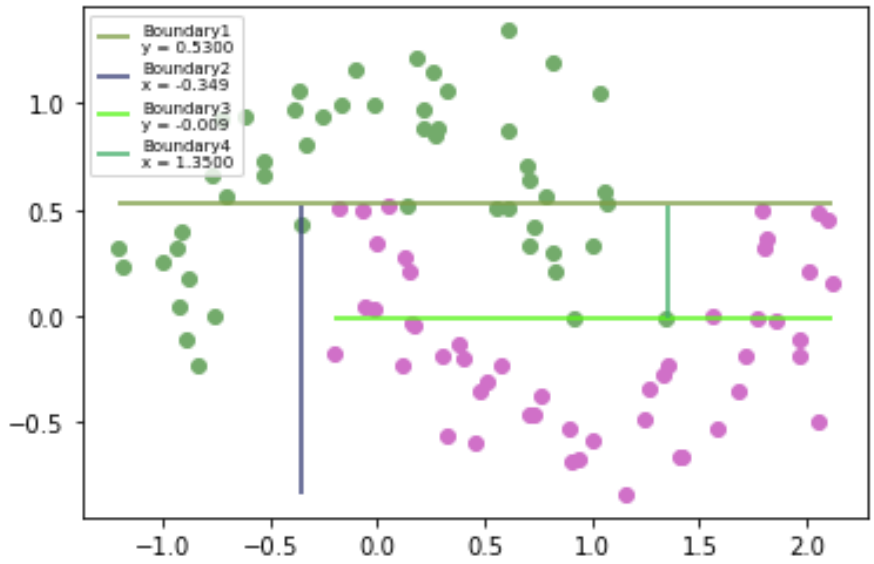
4.My own dataset:

4a.Data & Boundaries:

I created a dataset with make_moons (sklearn datasets).



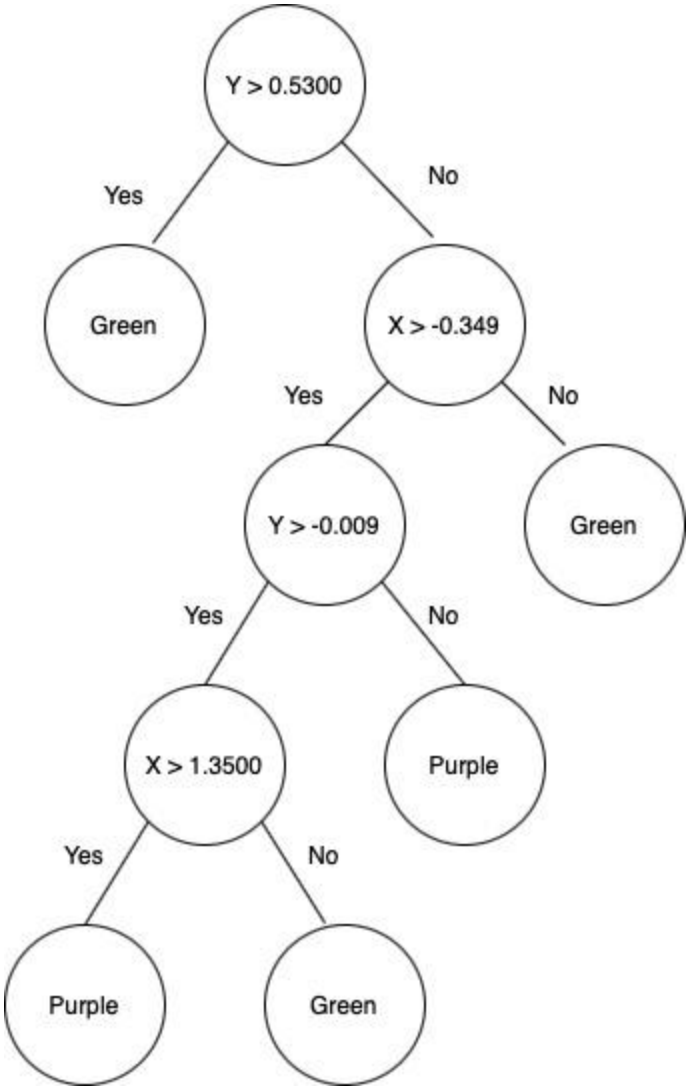
I used depth 4. My program can run for every depth value.



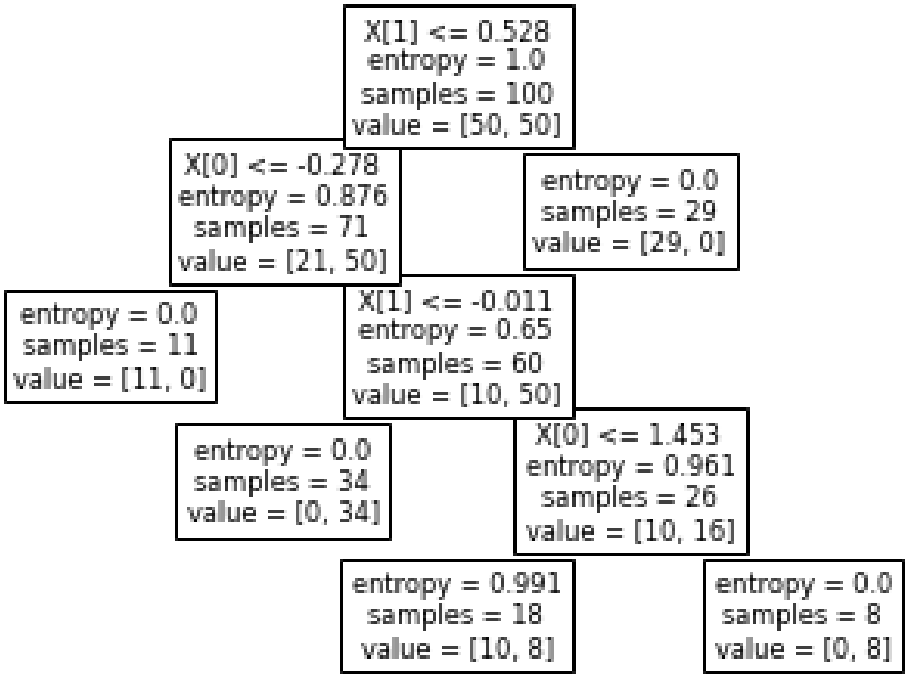
- Level 0 Entropy: 1.0
- Entropy of the left of level 0: 0.8760643678555243

- Entropy of the right of level 0: 0.0
- Weighted Entropy of level 0: 0.6220057011774223
- Entropy of the left of level 1: 0.0
- Entropy of the right of level 1: 0.6500224216483541
- Weighted Entropy of level 1: 0.5493147225197359
- Entropy of the left of level 2: 0.0
- Entropy of the right of level 2: 0.961236604722876
- Weighted Entropy of level 2: 0.4165358620465796
- Entropy of the left of level 3: 0.9910760598382222
- Entropy of the right of level 3: 0.0
- Weighted Entropy of level 3: 0.6861295798879999

4b.Decision Tree:



4c. Comparison with scikit-learn:



- Boundary 1: 0.528 Entropy : 1.0
- Boundary 2: -0.278 Entropy: 0.876
- Boundary 3: - 0.011 Entropy: 0.65
- Boundary 4: 1.453 Entropy: 0.961
- final left entropy : 0.991

My boundaries:

- Boundary 1: 0.5300 Entropy 1.0
- Boundary 2: -0.349 Entropy: 0.876
- Boundary 3: -0.009 Entropy: 0.65
- Boundary 4: 1.3500 Entropy: 0.961
- final left entropy: 0.991

Here again, I have the same Entropy values. However, boundary locations are slightly different, which occurs because my function iterates with 0.01 steps.