Optimal decision tree pipeline: proof of concept.

Phases:

1) Maximal hyperbox clustering

2) Minimal cutset

3) Minimal decision tree

So far five testsets: four tiny artificial (few points, 2 dimensions), fifth is iris dataset (https://archive.ics.uci.edu/dataset/53/iris), attributes only Sepal Width and Petal Length, well known to separate iris setosa from the other two species (trivial).

Phase 1 (optional), AABB clustering

Identifies AABB (hyperbox) clusters of points belonging to one same class. Reasonable when points are not too intermixed (which would make clustering meaningless, and the whole analysis of little relevance).

Directory hyperbox, two implementations: C++ (complete), python (incomplete, but produces plots).

C++: testset file must be written inside the code, file hyperbox.cpp.

Results will be in file hyperboxes_TESTSETNAME.txt

Phase 2: minimal cutset identification

Identifies the minimal set of cuts that separates exactly all dataset points.

Directory MIPmodel, two implementations: python starts from hyperboxes, c++ only uses only the points.

Python: flag fGoFromScratch, if true it generates internally the hyperboxes, if false it reads them (from the output of hyperboxes). The list of hyperboxes is different (internal and external boxes if generated, the real hyperboxes if read) In both cases the MIP model is the same SCP. Testset file must be written inside the code.

Results will be in file cuts_TESTSETNAME.txt

C++: it generates the cuts directly from the points, without previous hyperbox identification. A cut in every dimension when two points of different classes follow one another. Possibly many cuts, criss-cross (cut-column) generation. Still to be completed.

Phase 3: decision tree construction.

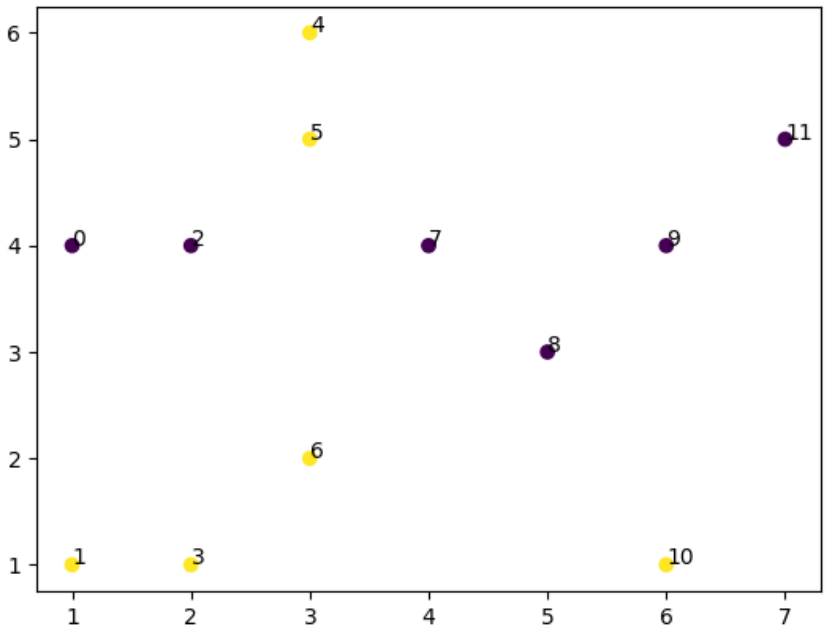
Given the cuts (easily if-then rules) generates a compatible decision tree. Max entropy criterion for node selection.

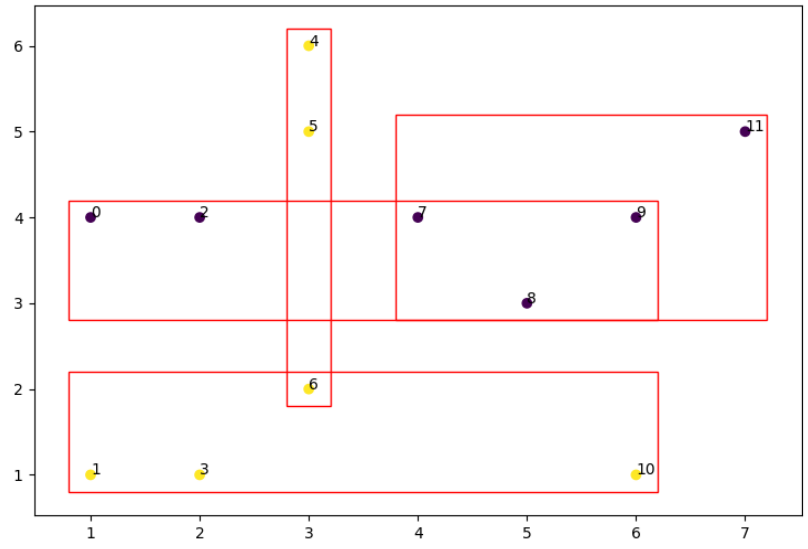
Directory plotTree, only c++ implementation.

It implements a DFS, actually a depth first construction, in the space of the regions identified by the cuts. At each iteration the entropy of each cut is computed on the (remaining) point set, max entropy gets the node. The tree is visualized by graphviz, externally.

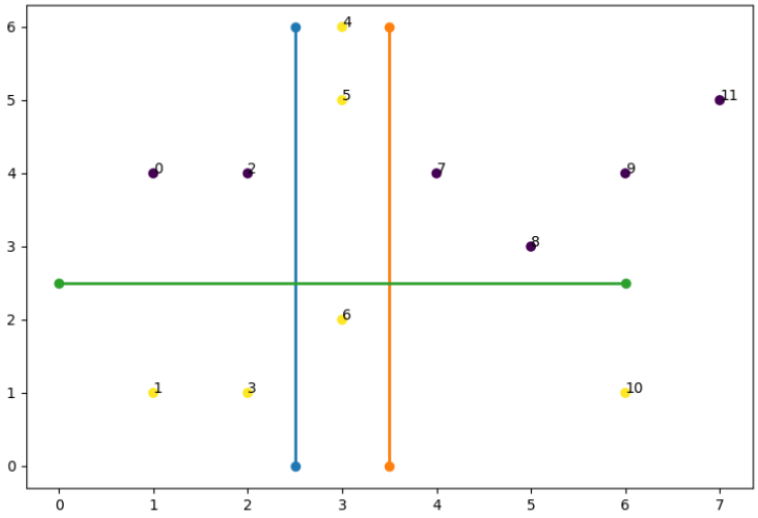
Bitmaps for the regions and for the stack of cuts used at each node. CANNOT EXCEED THE INT BITS!

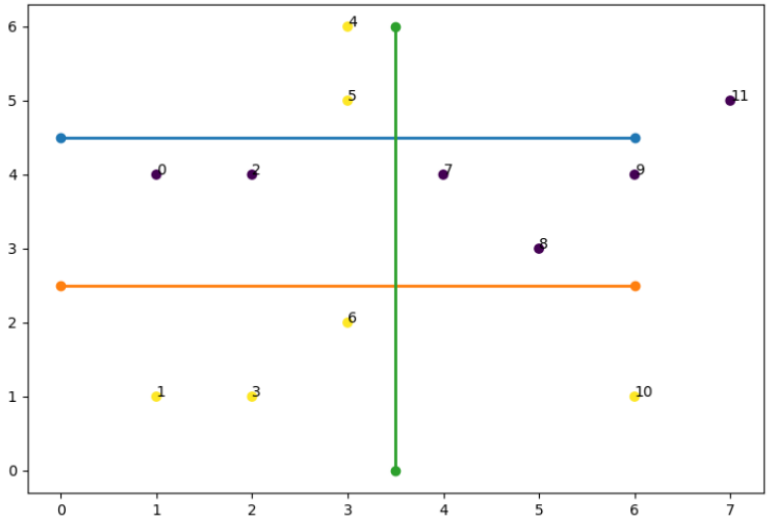
Test1

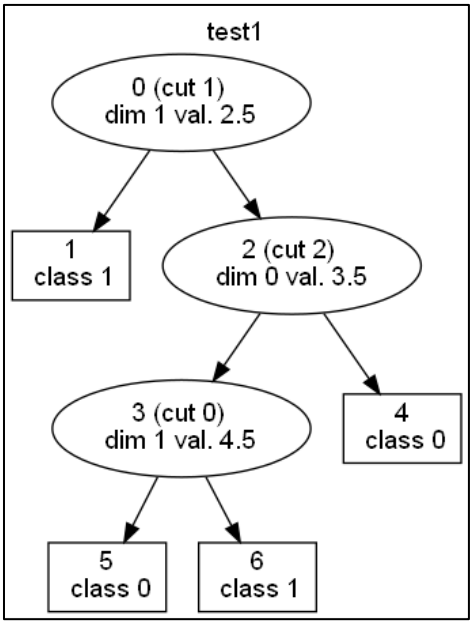




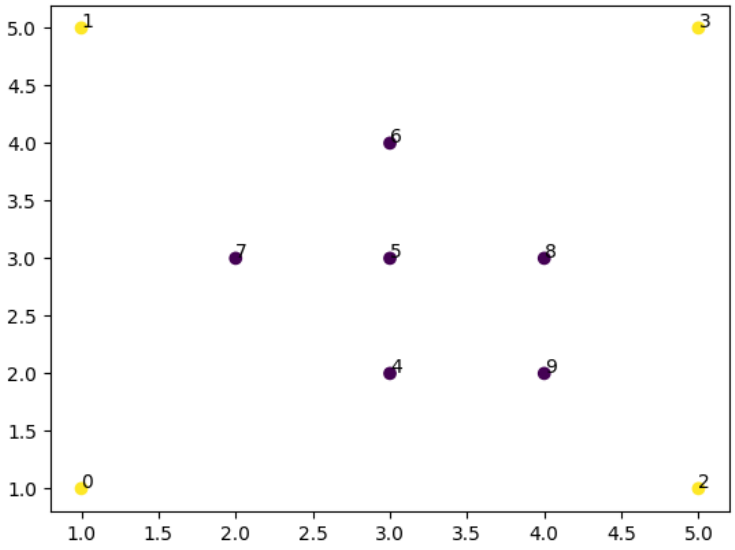
Two different (equivalent) cut selections when starting from scratch or from reading hyperboxes.

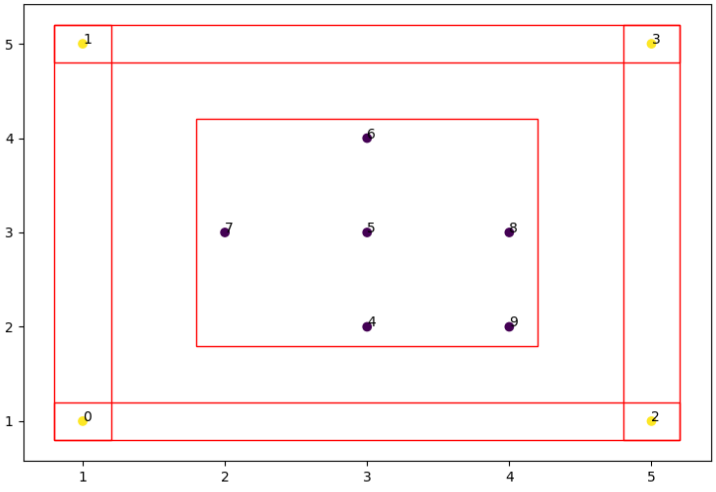

0
1
2
3
4
5
6
7
8
9
10
11

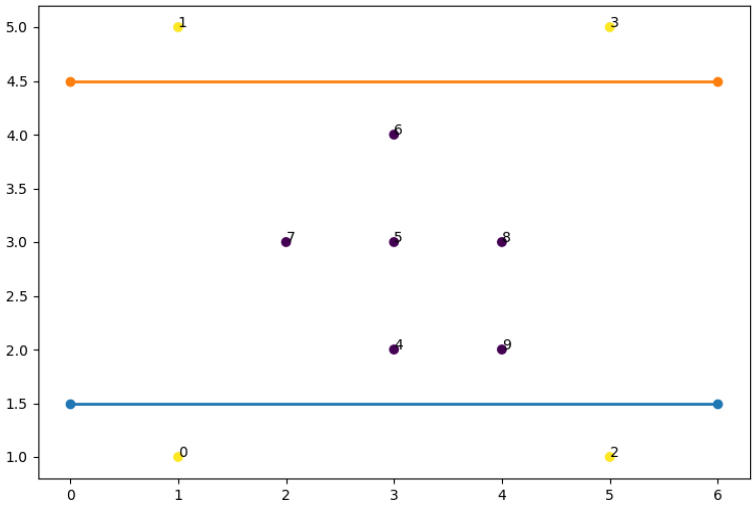

0
1
2
3
4
5
6
7
8
9
10
11

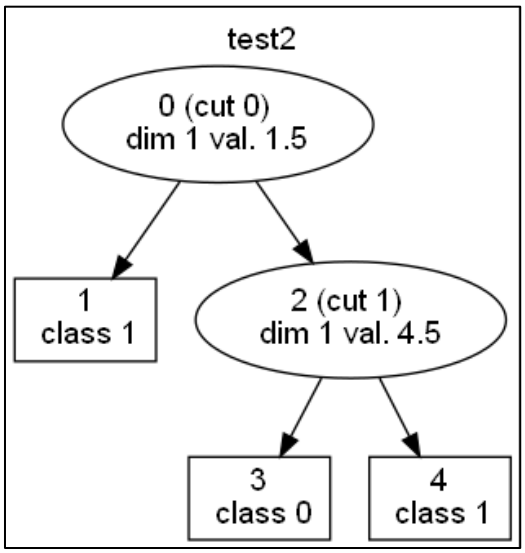

test1
0 (cut 1)
dim 1 val. 2.5
1
class 1
2 (cut 2)
dim 0 val. 3.5
3 (cut 0)
dim 1 val. 4.5
4
class 0
5
class 0
6
class 1

Test2

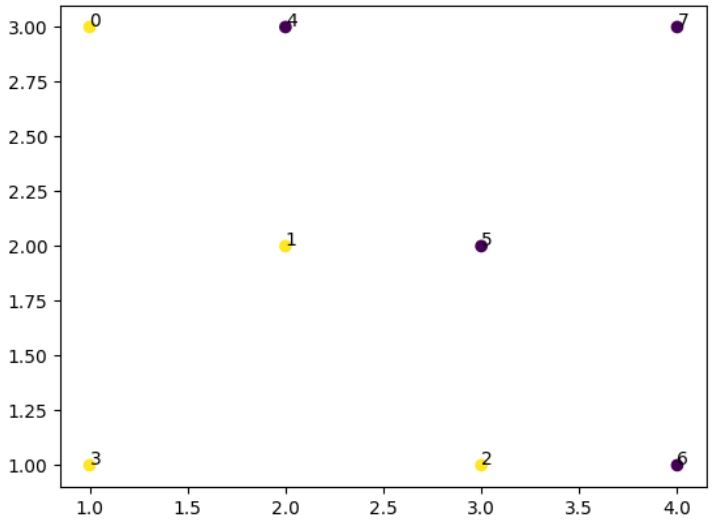

1
3
6
7
5
8
4
9
0
2
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

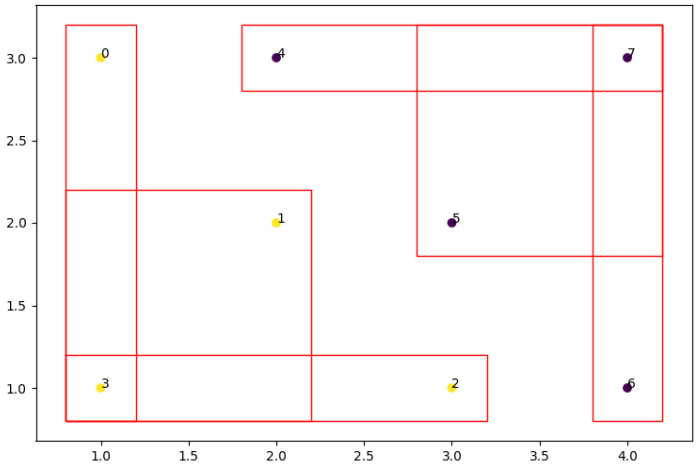

1
3
6
7
5
8
4
9
0
2
5
4
3
2
1
1
2
3
4
5

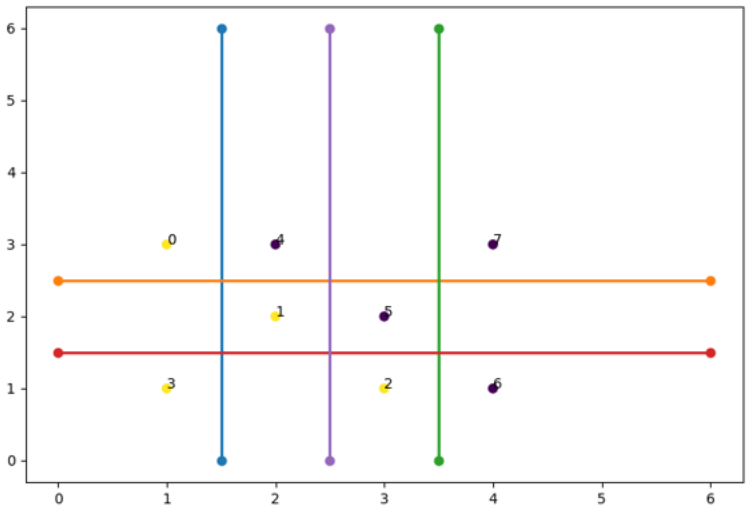

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
1
2
3
4
5
6
0
1
2
3
4
5
6
7
8
9

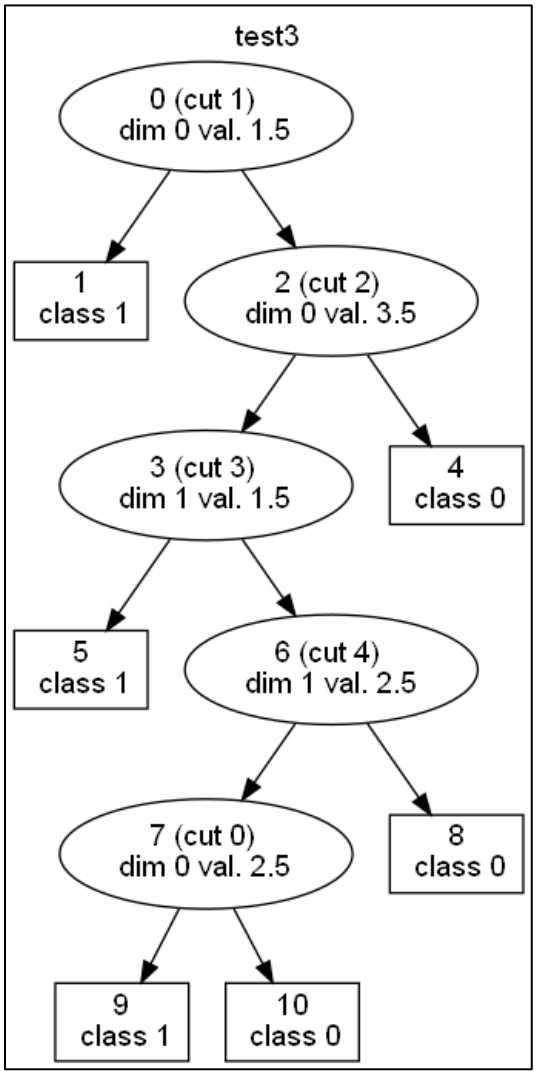

test2
0 (cut 0)
dim 1 val. 1.5
1
class 1
2 (cut 1)
dim 1 val. 4.5
3
class 0
4
class 1

Test 3

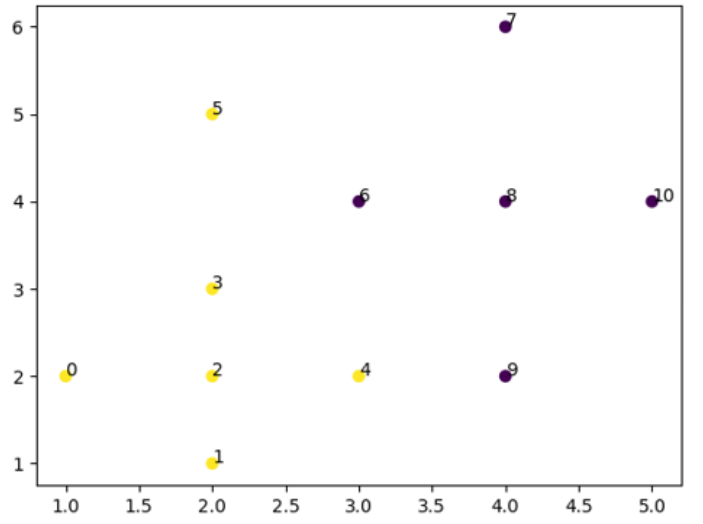

0
4
7
1
5
3
2
6
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
1.0
1.5
2.0
2.5
3.0
3.5
4.0

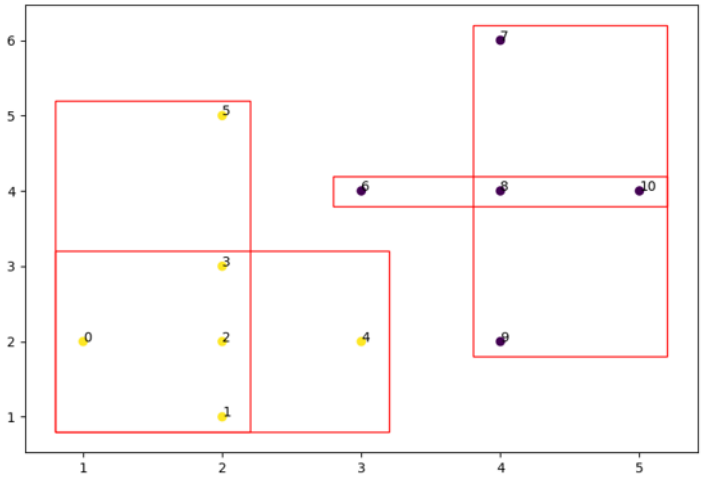

0
4
7
1
5
3
2
6
3.0
2.5
2.0
1.5
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0


0
4
7
1
5
3
2
6

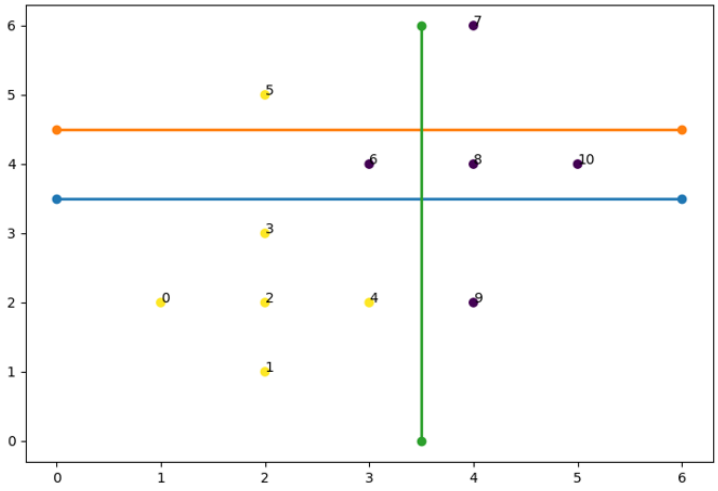

test3
0 (cut 1)
dim 0 val. 1.5
1
class 1
2 (cut 2)
dim 0 val. 3.5
3 (cut 3)
dim 1 val. 1.5
4
class 0
5
class 1
6 (cut 4)
dim 1 val. 2.5
7 (cut 0)
dim 0 val. 2.5
8
class 0
9
class 1
10
class 0

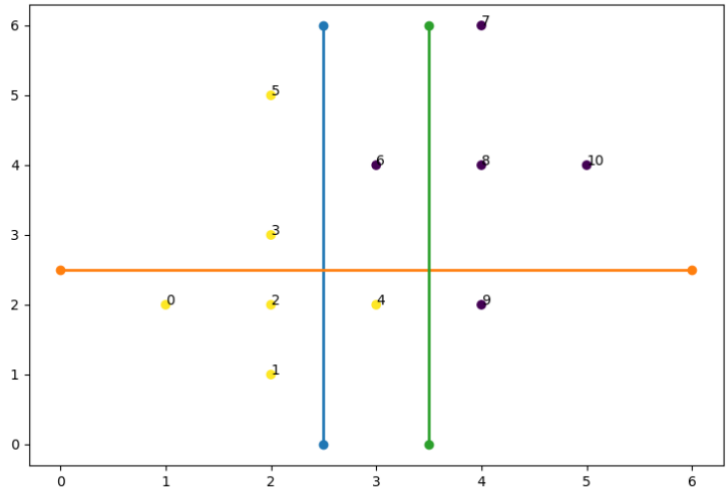
Test 4

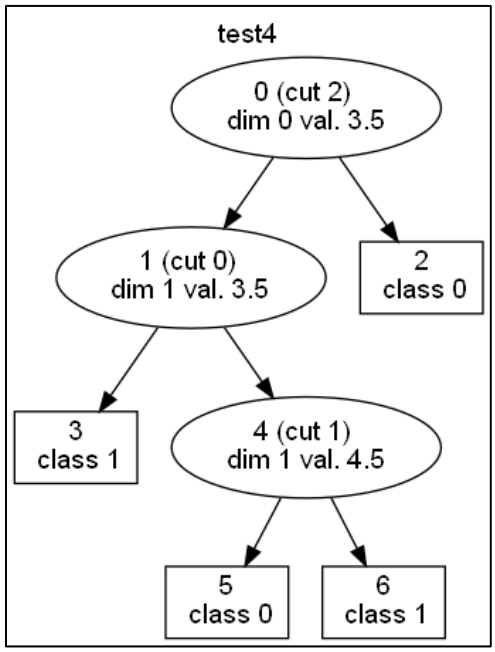




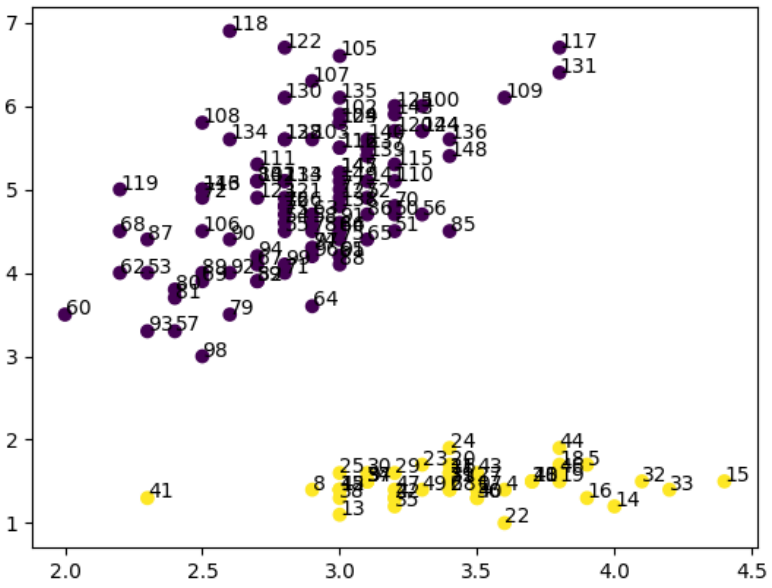
Two different (equivalent) cut selections when starting from scratch or from reading hyperboxes.

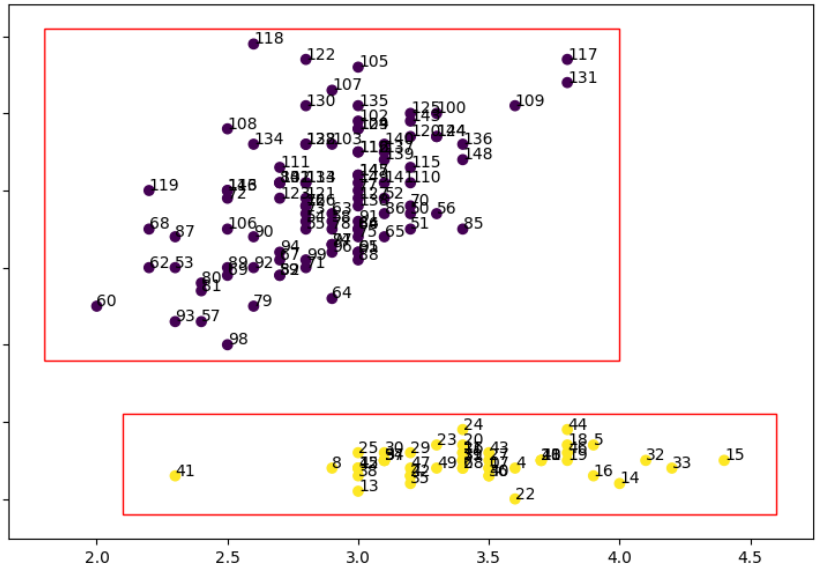

0
1
2
3
4
5
6
7
8
9
10

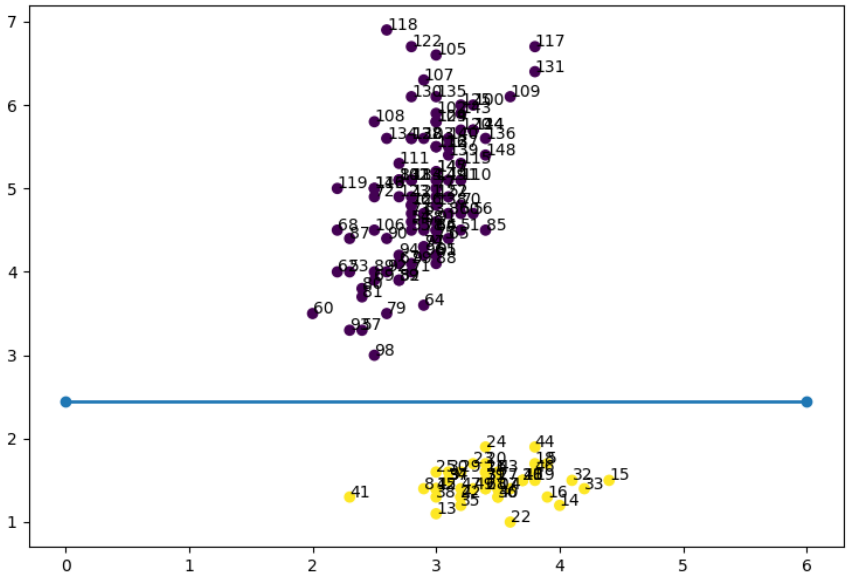

0
1
2
3
4
5
6
7
8
9
10

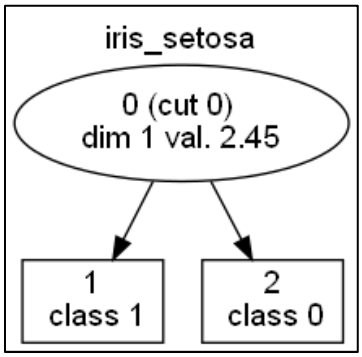

test4
0 (cut 2)
dim 0 val. 3.5
1 (cut 0)
dim 1 val. 3.5
2
class 0
3
class 1
4 (cut 1)
dim 1 val. 4.5
5
class 0
6
class 1

Iris, dimensions: SepalWidthCm, PetalLengthCm








iris_setosa
0 (cut 0)
dim 1 val. 2.45
1
class 1
2
class 0