ML LAB

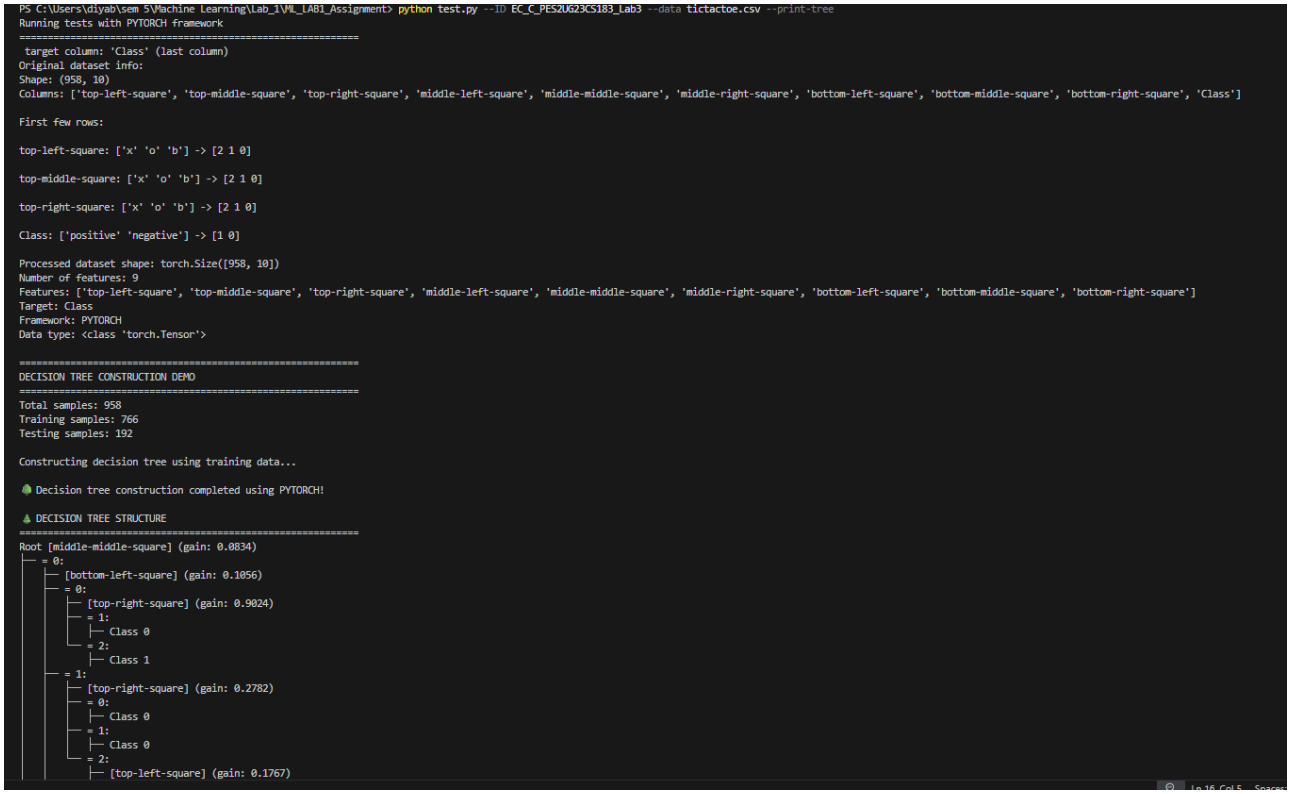
Name	Diya D Bhat
SRN	PES2UG23CS183
Section	C
Date	19/8/25

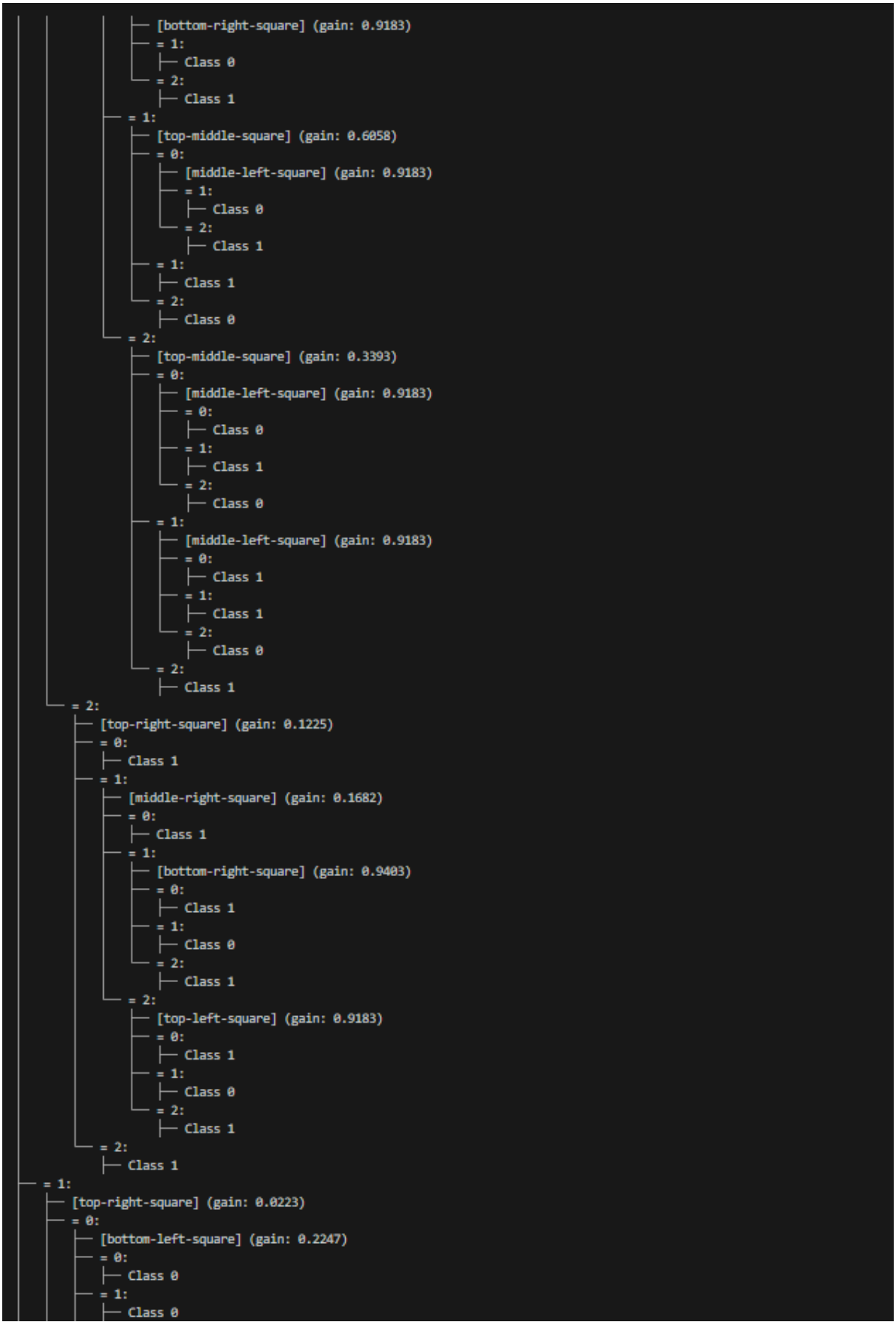
Dataset 1: Mushroom Classification

```
PS C:\Users\diyab\sem 5\Machine Learning\Lab_1\ML_LAB1_Assignment> python test.py --ID EC_C_PES2UG23CS183_Lab3 --data mushrooms.csv --print-tree
Running tests with PYTORCH framework
============================================================
 target column: 'class' (last column)
Original dataset info:
Shape: (8124, 23)
Columns: ['cap-shape', 'cap-surface', 'cap-color', 'bruises', 'odor', 'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color', 'stalk-shape', 'stalk-root',
 'stalk-surface-above-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-typ
e', 'spore-print-color', 'population', 'habitat', 'class']

First few rows:

cap-shape: ['x' 'b' 's' 'f' 'k'] -> [5 0 4 2 3]

cap-surface: ['s' 'y' 'f' 'g'] -> [2 3 0 1]

cap-color: ['n' 'y' 'w' 'g' 'e'] -> [4 9 8 3 2]

class: ['p' 'e'] -> [1 0]

Processed dataset shape: torch.Size([8124, 23])
Number of features: 22
Features: ['cap-shape', 'cap-surface', 'cap-color', 'bruises', 'odor', 'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color', 'stalk-shape', 'stalk-root'
, 'stalk-surface-above-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-ty
pe', 'spore-print-color', 'population', 'habitat']
Target: class
Framework: PYTORCH
Data type: <class 'torch.Tensor'>

============================================================
DECISION TREE CONSTRUCTION DEMO
============================================================
Total samples: 8124
Training samples: 6499
Testing samples: 1625

Constructing decision tree using training data...
```

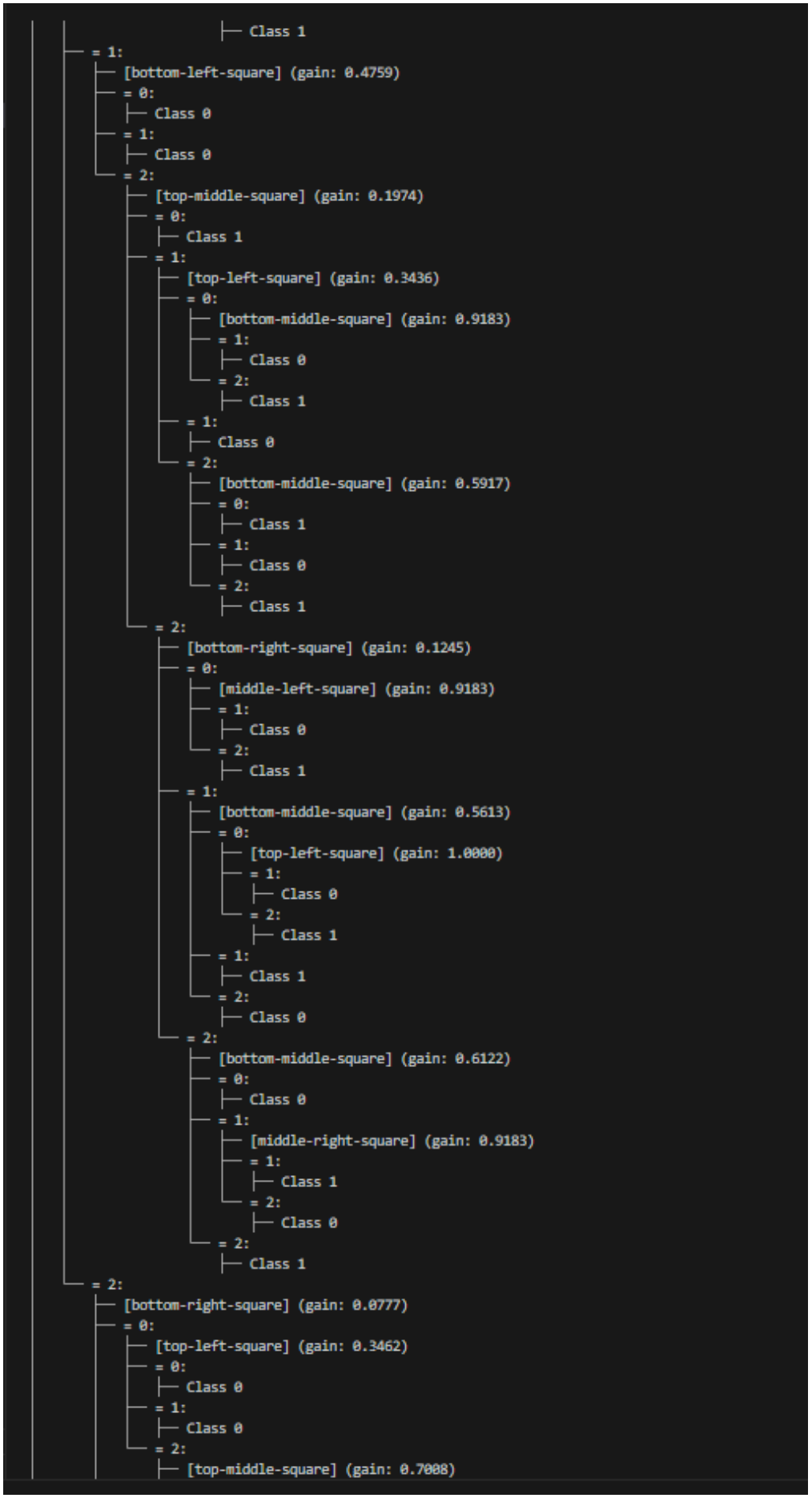
```
🌳 Decision tree construction completed using PYTORCH!

🌲 DECISION TREE STRUCTURE
============================================================
Root [odor] (gain: 0.9083)
├── = 0:
│   ├── Class 0
├── = 1:
│   ├── Class 1
├── = 2:
│   ├── Class 1
├── = 3:
│   ├── Class 0
├── = 4:
│   ├── Class 1
├── = 5:
│   ├── [spore-print-color] (gain: 0.1469)
│   ├── = 0:
│   │   ├── Class 0
│   ├── = 1:
│   │   ├── Class 0
│   ├── = 2:
│   │   ├── Class 0
│   ├── = 3:
│   │   ├── Class 0
│   ├── = 4:
│   │   ├── Class 0
│   ├── = 5:
│   │   ├── Class 1
│   ├── = 7:
│   │   ├── [habitat] (gain: 0.2217)
│   │   ├── = 0:
│   │   │   ├── [gill-size] (gain: 0.7642)
│   │   │   ├── = 0:
│   │   │   │   ├── Class 0
│   │   │   └── = 1:
│   │   │       ├── Class 1
│   │   ├── = 1:
│   │   │   ├── Class 0
│   │   ├── = 2:
│   │   │   ├── [cap-color] (gain: 0.7300)
│   │   │   ├── = 1:
│   │   │   │   ├── Class 0
│   │   │   ├── = 4:
│   │   │   │   ├── Class 0
│   │   │   ├── = 8:
│   │   │   │   ├── Class 1
│   │   │   └── = 9:
│   │   │       ├── Class 1
│   │   ├── = 4:
│   │   │   ├── Class 0
│   │   └── = 6:
│   │       ├── Class 0
│   └── = 8:
│       ├── Class 0
├── = 6:
│   ├── Class 1
├── = 7:
│   ├── Class 1
├── = 8:
│   ├── Class 1


📊 OVERALL PERFORMANCE METRICS
========================================
Accuracy:             1.0000 (100.00%)
Precision (weighted): 1.0000
Recall (weighted):    1.0000
F1-Score (weighted):  1.0000
Precision (macro):    1.0000
Recall (macro):       1.0000
F1-Score (macro):     1.0000

🌳 TREE COMPLEXITY METRICS
========================================
Maximum Depth:        4
Total Nodes:          29
Leaf Nodes:           24
Internal Nodes:       5
```

Dataset 2: Tic-Tac-Toe Endgame




[bottom-right-square] (gain: 0.9183)
= 1:
Class 0
= 2:
Class 1
= 1:
[top-middle-square] (gain: 0.6058)
= 0:
[middle-left-square] (gain: 0.9183)
= 1:
Class 0
= 2:
Class 1
= 1:
Class 1
= 2:
Class 0
= 2:
[top-middle-square] (gain: 0.3393)
= 0:
[middle-left-square] (gain: 0.9183)
= 0:
Class 0
= 1:
Class 1
= 2:
Class 0
= 1:
[middle-left-square] (gain: 0.9183)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 0
= 2:
Class 1
= 2:
[top-right-square] (gain: 0.1225)
= 0:
Class 1
= 1:
[middle-right-square] (gain: 0.1682)
= 0:
Class 1
= 1:
[bottom-right-square] (gain: 0.9403)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 1
= 2:
[top-left-square] (gain: 0.9183)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 1
= 2:
Class 1
= 1:
[top-right-square] (gain: 0.0223)
= 0:
[bottom-left-square] (gain: 0.2247)
= 0:
Class 0
= 1:
Class 0


= 1:
[bottom-right-square] (gain: 0.9403)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 1
= 2:
[top-left-square] (gain: 0.9183)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 1
= 2:
Class 1
= 1:
[top-right-square] (gain: 0.0223)
= 0:
[bottom-left-square] (gain: 0.2247)
= 0:
Class 0
= 1:
Class 0
= 2:
[middle-right-square] (gain: 0.1159)
= 0:
[top-left-square] (gain: 0.1771)
= 0:
[middle-left-square] (gain: 0.9183)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 0
= 1:
[bottom-right-square] (gain: 0.9710)
= 1:
Class 0
= 2:
Class 1
= 2:
Class 1
= 1:
[middle-left-square] (gain: 0.9887)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 1
= 2:
[bottom-middle-square] (gain: 0.2400)
= 0:
[top-left-square] (gain: 1.0000)
= 1:
Class 0
= 2:
Class 1
= 1:
Class 0
= 2:
[bottom-right-square] (gain: 0.9710)
= 1:
Class 0
= 2:
Class 1


Class 1
= 1:
[bottom-left-square] (gain: 0.4759)
= 0:
Class 0
= 1:
Class 0
= 2:
[top-middle-square] (gain: 0.1974)
= 0:
Class 1
= 1:
[top-left-square] (gain: 0.3436)
= 0:
[bottom-middle-square] (gain: 0.9183)
= 1:
Class 0
= 2:
Class 1
= 1:
Class 0
= 2:
[bottom-middle-square] (gain: 0.5917)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 1
= 2:
[bottom-right-square] (gain: 0.1245)
= 0:
[middle-left-square] (gain: 0.9183)
= 1:
Class 0
= 2:
Class 1
= 1:
[bottom-middle-square] (gain: 0.5613)
= 0:
[top-left-square] (gain: 1.0000)
= 1:
Class 0
= 2:
Class 1
= 1:
Class 1
= 2:
Class 0
= 2:
[bottom-middle-square] (gain: 0.6122)
= 0:
Class 0
= 1:
[middle-right-square] (gain: 0.9183)
= 1:
Class 1
= 2:
Class 0
= 2:
Class 1
= 2:
[bottom-right-square] (gain: 0.0777)
= 0:
[top-left-square] (gain: 0.3462)
= 0:
Class 0
= 1:
Class 0
= 2:
[top-middle-square] (gain: 0.7008)

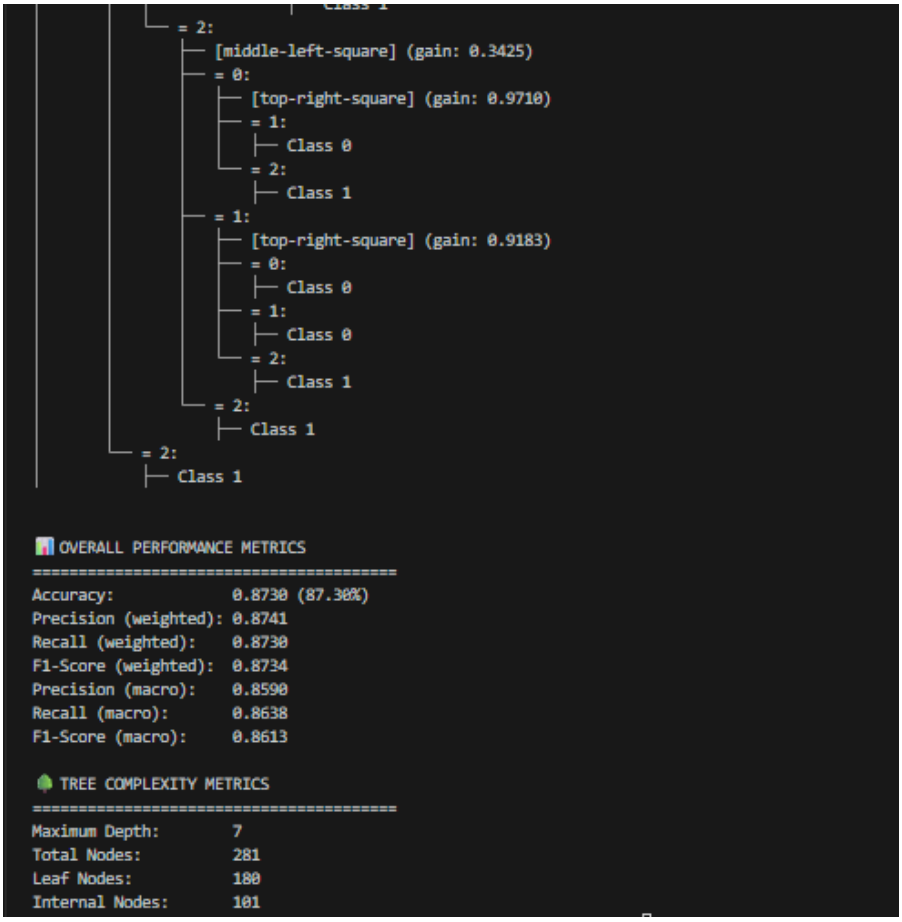

= 0:
Class 0
= 1:
[middle-right-square] (gain: 0.7219)
= 0:
Class 0
= 1:
Class 1
= 2:
Class 0
= 2:
Class 1
= 1:
[top-left-square] (gain: 0.5439)
= 0:
Class 0
= 1:
Class 0
= 2:
[top-middle-square] (gain: 0.4687)
= 0:
[bottom-middle-square] (gain: 0.9183)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 0
= 1:
[middle-right-square] (gain: 0.9183)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 0
= 2:
Class 1
= 2:
[middle-right-square] (gain: 0.4731)
= 0:
[top-middle-square] (gain: 0.6464)
= 0:
Class 1
= 1:
Class 0
= 2:
[top-left-square] (gain: 0.8113)
= 1:
Class 0
= 2:
Class 1
= 1:
[middle-left-square] (gain: 0.3995)
= 0:
[bottom-middle-square] (gain: 0.8113)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 1
= 1:
Class 0
= 2:
[top-middle-square] (gain: 0.8113)
= 1:
Class 0
= 2:
Class 1


= 2:
Class 1
= 2:
[bottom-right-square] (gain: 0.0269)
= 0:
[top-left-square] (gain: 0.1239)
= 0:
Class 1
= 1:
[bottom-middle-square] (gain: 0.1033)
= 0:
[middle-left-square] (gain: 0.1605)
= 0:
Class 1
= 1:
[bottom-left-square] (gain: 1.0000)
= 1:
Class 0
= 2:
Class 1
= 2:
[middle-right-square] (gain: 0.5917)
= 0:
Class 0
= 1:
Class 1
= 2:
Class 1
= 1:
Class 1
= 2:
[top-middle-square] (gain: 0.4591)
= 0:
[middle-right-square] (gain: 0.9183)
= 0:
Class 0
= 1:
Class 1
= 2:
Class 0
= 1:
[top-right-square] (gain: 0.6122)
= 0:
Class 1
= 1:
Class 0
= 2:
[middle-right-square] (gain: 0.9183)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 0
= 2:
Class 1
= 2:
Class 1
= 1:
[top-left-square] (gain: 0.0713)
= 0:
[middle-right-square] (gain: 0.0760)
= 0:
[bottom-left-square] (gain: 0.2455)
= 0:
Class 1
= 1:
[bottom-middle-square] (gain: 0.9710)
= 1:
Class 0
= 2:


Class 1
= 2:
Class 1
= 1:
[top-right-square] (gain: 0.7207)
= 0:
Class 1
= 1:
Class 0
= 2:
[middle-left-square] (gain: 0.3060)
= 0:
Class 1
= 1:
Class 1
= 2:
[top-middle-square] (gain: 1.0000)
= 0:
Class 1
= 2:
Class 0
= 2:
[middle-left-square] (gain: 0.2294)
= 0:
[top-middle-square] (gain: 0.5000)
= 0:
Class 0
= 1:
Class 1
= 2:
[top-right-square] (gain: 1.0000)
= 0:
Class 0
= 1:
Class 1
= 1:
[bottom-left-square] (gain: 0.3219)
= 0:
Class 1
= 1:
[top-right-square] (gain: 1.0000)
= 0:
Class 1
= 2:
Class 0
= 2:
Class 1
= 2:
Class 1
= 1:
[bottom-left-square] (gain: 0.1145)
= 0:
Class 1
= 1:
[middle-left-square] (gain: 0.3407)
= 0:
[bottom-middle-square] (gain: 0.9183)
= 1:
Class 0
= 2:
Class 1
= 1:
Class 0
= 2:
[bottom-middle-square] (gain: 0.6500)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 1


= 1:
Class 0
= 2:
Class 1
= 2:
[top-right-square] (gain: 0.1913)
= 0:
Class 1
= 1:
[top-middle-square] (gain: 0.3774)
= 0:
Class 1
= 1:
Class 0
= 2:
[middle-right-square] (gain: 0.8113)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 1
= 2:
Class 1
= 2:
[top-right-square] (gain: 0.0822)
= 0:
[bottom-left-square] (gain: 0.3436)
= 0:
Class 1
= 1:
[bottom-middle-square] (gain: 0.9852)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 1
= 2:
Class 1
= 1:
[middle-right-square] (gain: 0.4301)
= 0:
[bottom-middle-square] (gain: 0.9183)
= 0:
Class 1
= 1:
Class 0
= 2:
Class 1
= 1:
Class 0
= 2:
[middle-left-square] (gain: 0.6101)
= 0:
Class 0
= 1:
[bottom-left-square] (gain: 0.9183)
= 1:
Class 1
= 2:
Class 0
= 2:
Class 1
= 2:
[bottom-left-square] (gain: 0.5549)
= 0:
Class 1
= 1:
[top-middle-square] (gain: 0.8113)
= 0:

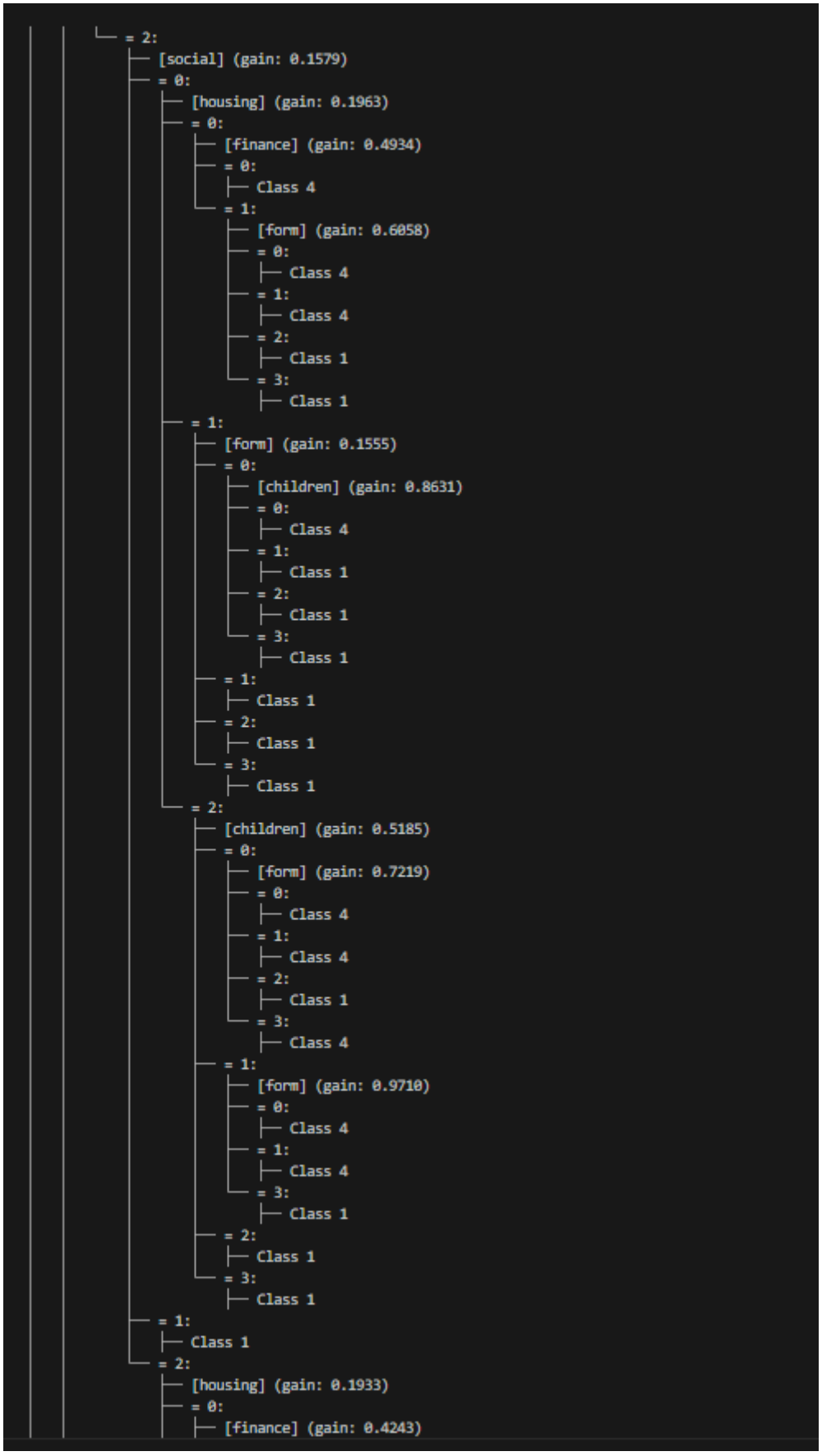
```
                ├── = 0:
                │   ├── Class 1
                ├── = 1:
                │   ├── [top-middle-square] (gain: 0.8113)
                │   ├── = 0:
                │   │   ├── Class 0
                │   ├── = 1:
                │   │   ├── Class 0
                │   └── = 2:
                │       ├── Class 1
                └── = 2:
                    ├── Class 1
    └── = 2:
        ├── [top-left-square] (gain: 0.2780)
        ├── = 0:
        │   ├── Class 1
        ├── = 1:
        │   ├── [middle-right-square] (gain: 0.0689)
        │   ├── = 0:
        │   │   ├── [bottom-left-square] (gain: 0.1985)
        │   │   ├── = 0:
        │   │   │   ├── Class 0
        │   │   ├── = 1:
        │   │   │   ├── [middle-left-square] (gain: 0.5917)
        │   │   │   ├── = 0:
        │   │   │   │   ├── Class 1
        │   │   │   ├── = 1:
        │   │   │   │   ├── Class 0
        │   │   │   └── = 2:
        │   │   │       ├── Class 0
        │   │   └── = 2:
        │   │       ├── [middle-left-square] (gain: 0.9710)
        │   │       ├── = 0:
        │   │       │   ├── Class 0
        │   │       ├── = 1:
        │   │       │   ├── Class 1
        │   │       └── = 2:
        │   │           ├── Class 0
        │   ├── = 1:
        │   │   ├── [middle-left-square] (gain: 0.4046)
        │   │   ├── = 0:
        │   │   │   ├── Class 1
        │   │   ├── = 1:
        │   │   │   ├── [bottom-left-square] (gain: 0.9183)
        │   │   │   ├── = 1:
        │   │   │   │   ├── Class 0
        │   │   │   └── = 2:
        │   │   │       ├── Class 1
        │   │   └── = 2:
        │   │       ├── [bottom-left-square] (gain: 0.0760)
        │   │       ├── = 0:
        │   │       │   ├── Class 0
        │   │       ├── = 1:
        │   │       │   ├── [top-right-square] (gain: 0.9183)
        │   │       │   ├── = 1:
        │   │       │   │   ├── Class 1
        │   │       │   └── = 2:
        │   │       │       ├── Class 0
        │   │       └── = 2:
        │   │           ├── [top-right-square] (gain: 0.9183)
        │   │           ├── = 1:
        │   │           │   ├── Class 0
        │   │           └── = 2:
        │   │               ├── Class 1
        │   └── = 2:
        │       ├── [middle-left-square] (gain: 0.3425)
        │       ├── = 0:
        │       │   ├── [top-right-square] (gain: 0.9710)
        │       │   ├── = 1:
        │       │   │   ├── Class 0
        │       │   └── = 2:
```

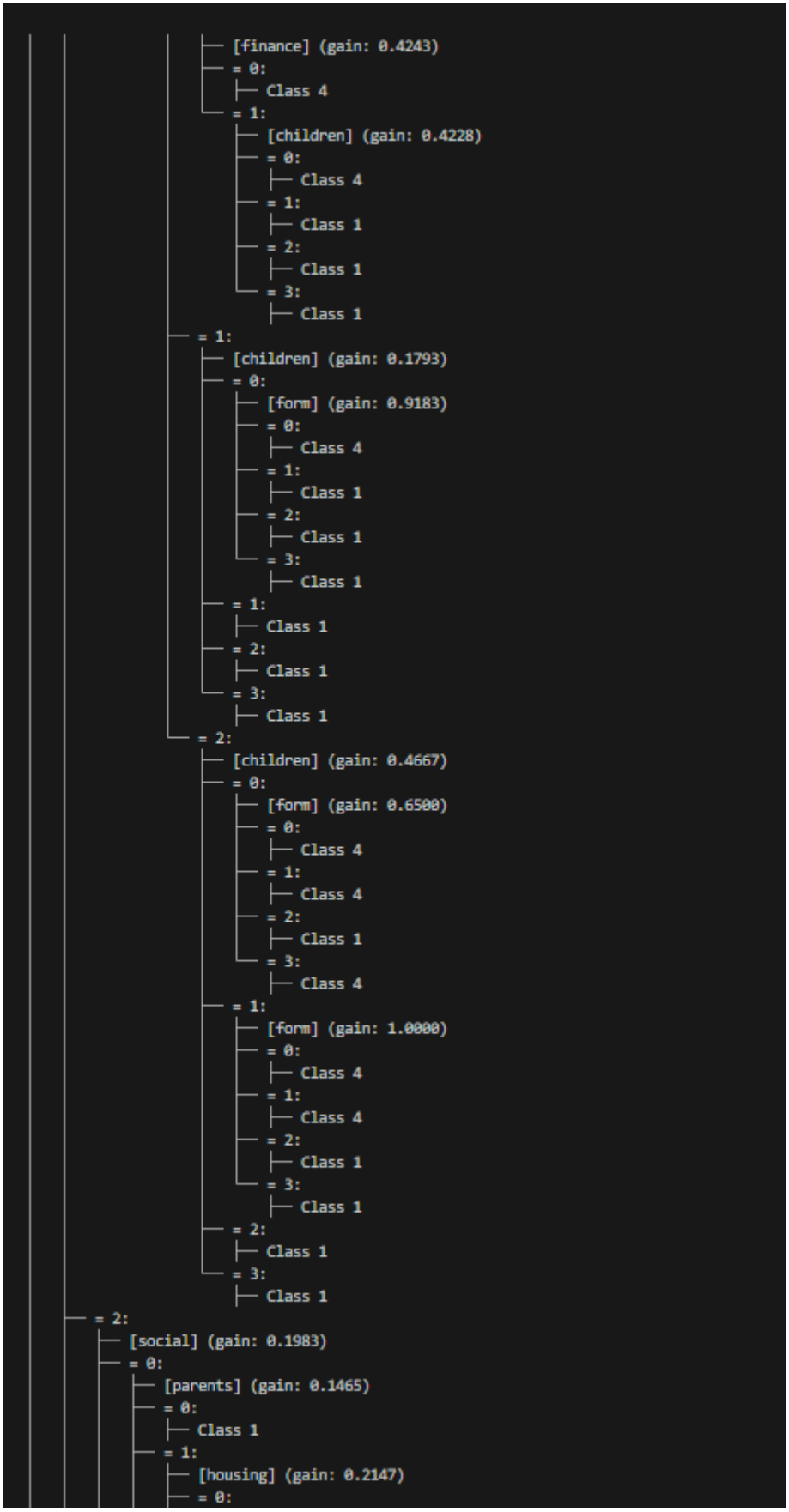


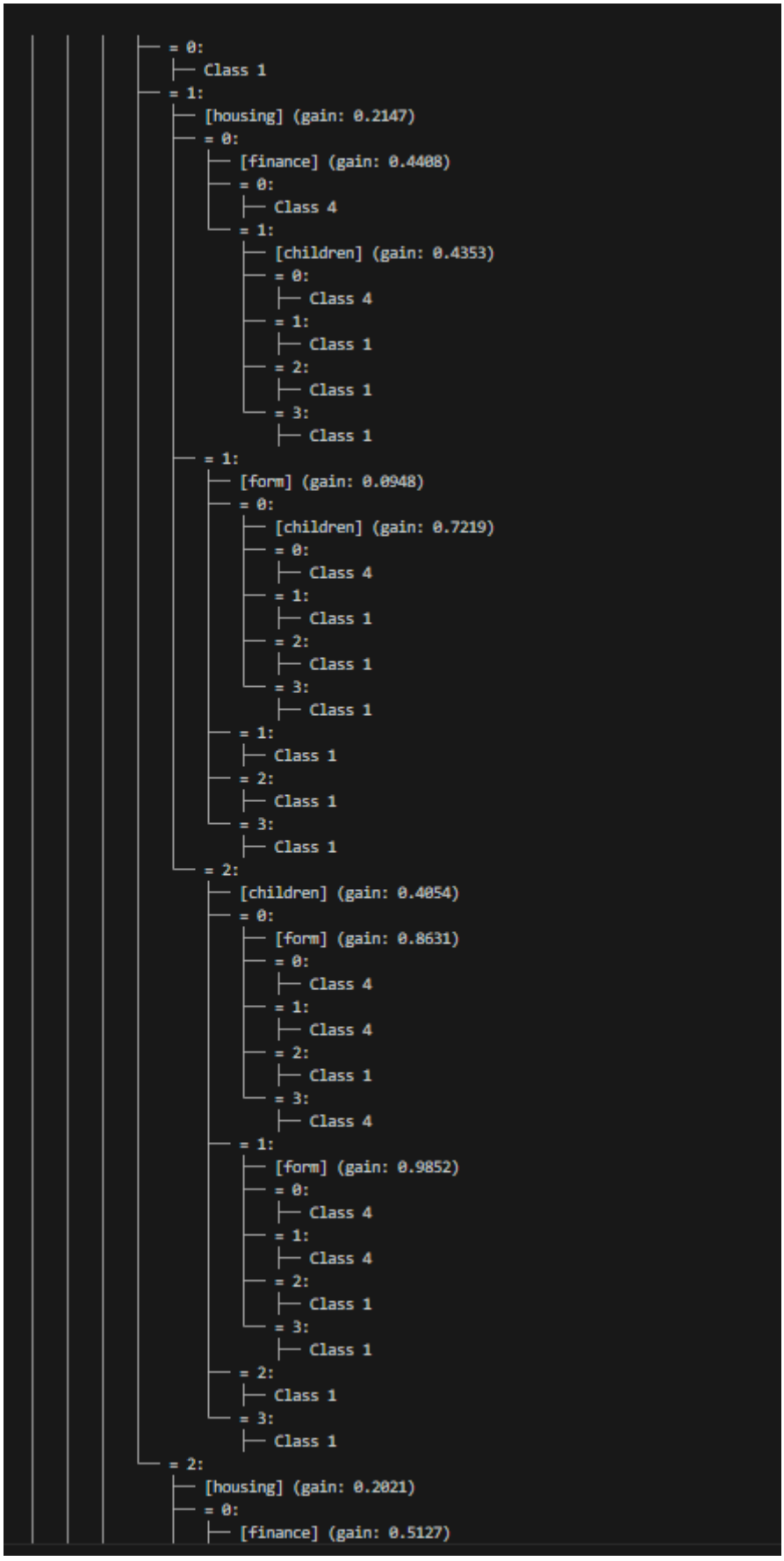
Dataset 3: Nursery School

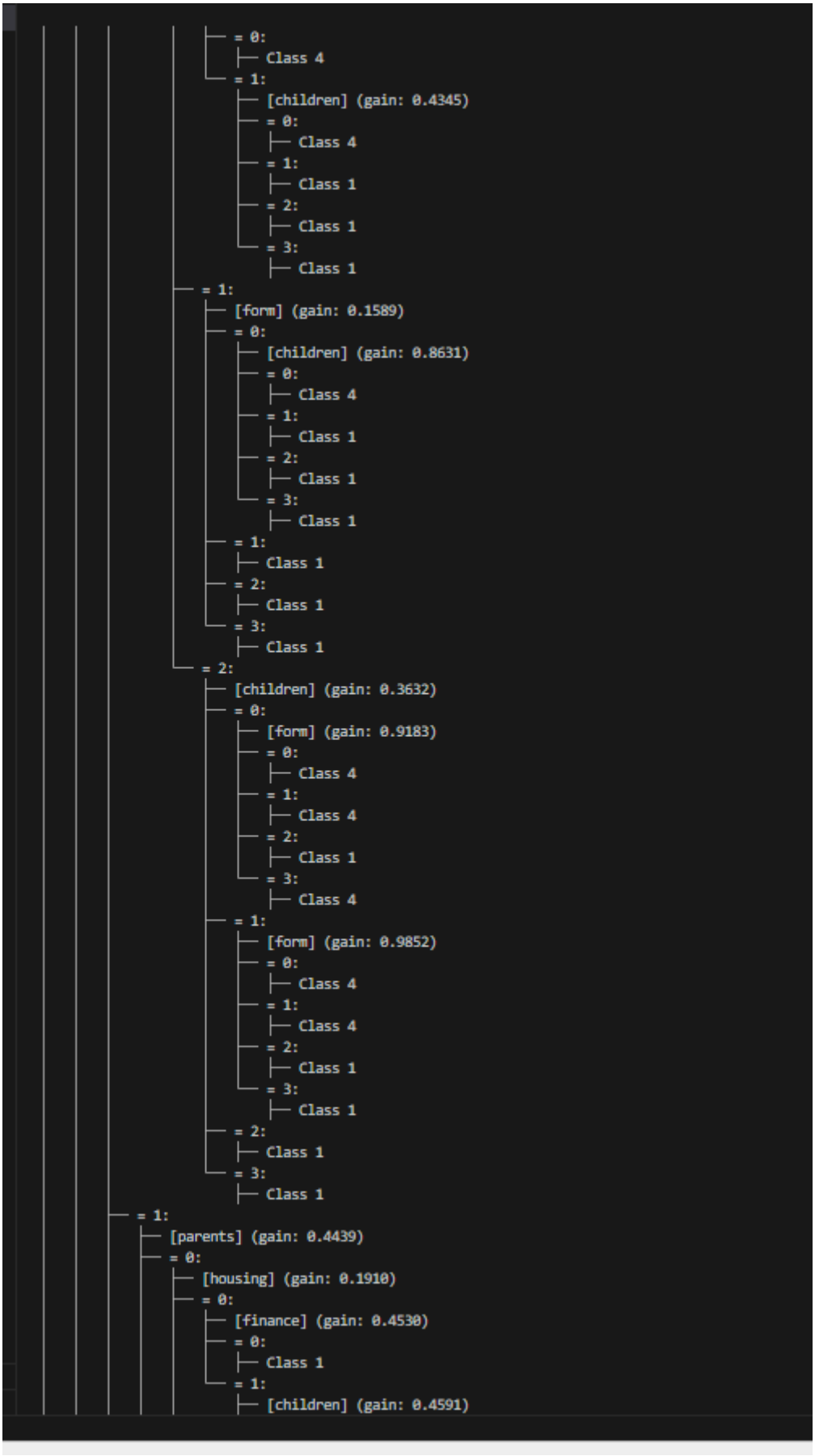
```
parents: ['usual' 'pretentious' 'great_pret'] -> [2 1 0]

has_nurs: ['proper' 'less_proper' 'improper' 'critical' 'very_crit'] -> [3 2 1 0 4]

form: ['complete' 'completed' 'incomplete' 'foster'] -> [0 1 3 2]

class: ['recommend' 'priority' 'not_recom' 'very_recom' 'spec_prior'] -> [2 1 0 4 3]

Processed dataset shape: torch.Size([12960, 9])
Number of features: 8
Features: ['parents', 'has_nurs', 'form', 'children', 'housing', 'finance', 'social', 'health']
Target: class
Framework: PYTORCH
Data type: <class 'torch.Tensor'>

============================================================
DECISION TREE CONSTRUCTION DEMO
============================================================
Total samples: 12960
Training samples: 10368
Testing samples: 2592

Constructing decision tree using training data...
```

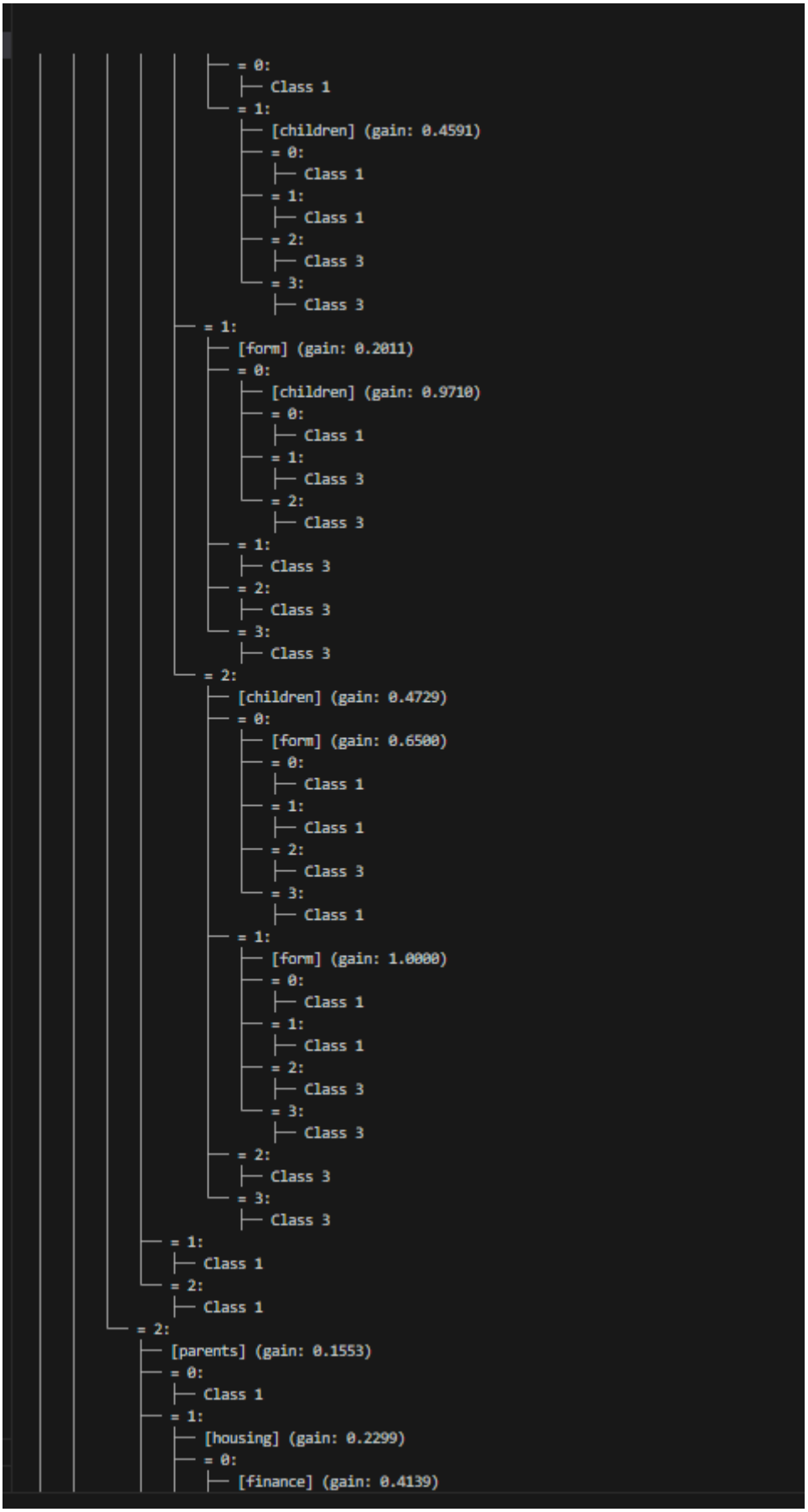

= 3:
Class 3
= 1:
[social] (gain: 0.4640)
= 0:
Class 1
= 1:
[housing] (gain: 0.1886)
= 0:
[finance] (gain: 0.5577)
= 0:
Class 1
= 1:
[form] (gain: 0.3555)
= 0:
Class 3
= 1:
Class 1
= 2:
Class 3
= 3:
Class 3
= 1:
[form] (gain: 0.1011)
= 0:
[children] (gain: 0.7219)
= 0:
Class 1
= 1:
Class 3
= 2:
Class 3
= 3:
Class 3
= 1:
Class 3
= 2:
Class 3
= 3:
Class 3
= 2:
[children] (gain: 0.5044)
= 0:
[form] (gain: 0.8113)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 3
= 1:
[form] (gain: 0.9183)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 3
= 3:
Class 3
= 2:

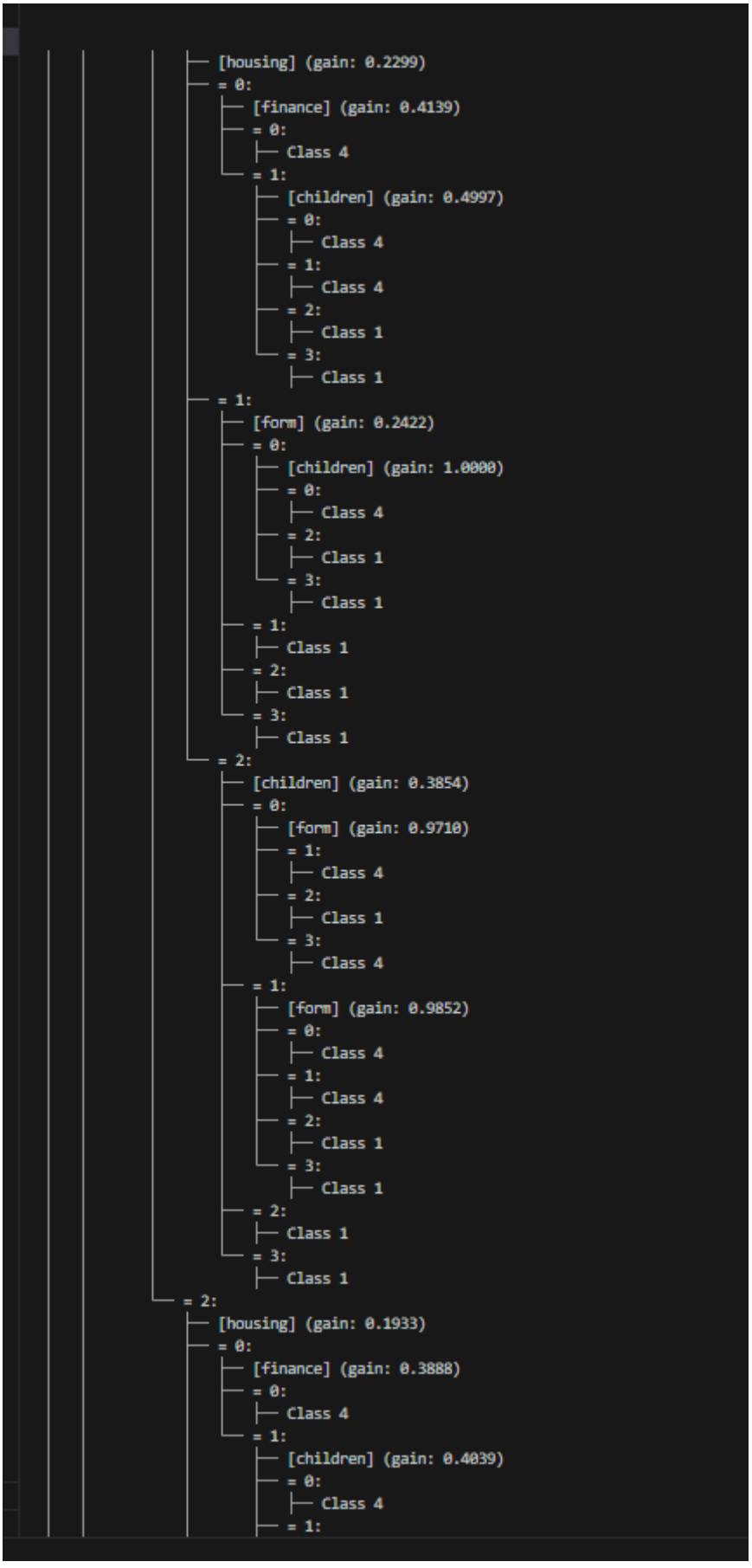

= 2:
[social] (gain: 0.1579)
= 0:
[housing] (gain: 0.1963)
= 0:
[finance] (gain: 0.4934)
= 0:
Class 4
= 1:
[form] (gain: 0.6058)
= 0:
Class 4
= 1:
Class 4
= 2:
Class 1
= 3:
Class 1
= 1:
[form] (gain: 0.1555)
= 0:
[children] (gain: 0.8631)
= 0:
Class 4
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 2:
[children] (gain: 0.5185)
= 0:
[form] (gain: 0.7219)
= 0:
Class 4
= 1:
Class 4
= 2:
Class 1
= 3:
Class 4
= 1:
[form] (gain: 0.9710)
= 0:
Class 4
= 1:
Class 4
= 3:
Class 1
= 2:
Class 1
= 3:
Class 1
= 1:
Class 1
= 2:
[housing] (gain: 0.1933)
= 0:
[finance] (gain: 0.4243)


[finance] (gain: 0.4243)
= 0:
Class 4
= 1:
[children] (gain: 0.4228)
= 0:
Class 4
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 1:
[children] (gain: 0.1793)
= 0:
[form] (gain: 0.9183)
= 0:
Class 4
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 2:
[children] (gain: 0.4667)
= 0:
[form] (gain: 0.6500)
= 0:
Class 4
= 1:
Class 4
= 2:
Class 1
= 3:
Class 4
= 1:
[form] (gain: 1.0000)
= 0:
Class 4
= 1:
Class 4
= 2:
Class 1
= 3:
Class 1
= 2:
Class 1
= 3:
Class 1
= 2:
[social] (gain: 0.1983)
= 0:
[parents] (gain: 0.1465)
= 0:
Class 1
= 1:
[housing] (gain: 0.2147)
= 0:


= 0:
Class 1
= 1:
[housing] (gain: 0.2147)
= 0:
[finance] (gain: 0.4408)
= 0:
Class 4
= 1:
[children] (gain: 0.4353)
= 0:
Class 4
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 1:
[form] (gain: 0.0948)
= 0:
[children] (gain: 0.7219)
= 0:
Class 4
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 2:
[children] (gain: 0.4054)
= 0:
[form] (gain: 0.8631)
= 0:
Class 4
= 1:
Class 4
= 2:
Class 1
= 3:
Class 4
= 1:
[form] (gain: 0.9852)
= 0:
Class 4
= 1:
Class 4
= 2:
Class 1
= 3:
Class 1
= 2:
Class 1
= 3:
Class 1
= 2:
[housing] (gain: 0.2021)
= 0:
[finance] (gain: 0.5127)


= 0:
Class 4
= 1:
[children] (gain: 0.4345)
= 0:
Class 4
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 1:
[form] (gain: 0.1589)
= 0:
[children] (gain: 0.8631)
= 0:
Class 4
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 2:
[children] (gain: 0.3632)
= 0:
[form] (gain: 0.9183)
= 0:
Class 4
= 1:
Class 4
= 2:
Class 1
= 3:
Class 4
= 1:
[form] (gain: 0.9852)
= 0:
Class 4
= 1:
Class 4
= 2:
Class 1
= 3:
Class 1
= 2:
Class 1
= 3:
Class 1
= 1:
[parents] (gain: 0.4439)
= 0:
[housing] (gain: 0.1910)
= 0:
[finance] (gain: 0.4530)
= 0:
Class 1
= 1:
[children] (gain: 0.4591)


= 0:
Class 1
= 1:
[children] (gain: 0.4591)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 3
= 3:
Class 3
= 1:
[form] (gain: 0.2011)
= 0:
[children] (gain: 0.9710)
= 0:
Class 1
= 1:
Class 3
= 2:
Class 3
= 1:
Class 3
= 2:
Class 3
= 3:
Class 3
= 2:
[children] (gain: 0.4729)
= 0:
[form] (gain: 0.6500)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 3
= 3:
Class 1
= 1:
[form] (gain: 1.0000)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 3
= 3:
Class 3
= 2:
Class 3
= 3:
Class 3
= 1:
Class 1
= 2:
Class 1
= 2:
[parents] (gain: 0.1553)
= 0:
Class 1
= 1:
[housing] (gain: 0.2299)
= 0:
[finance] (gain: 0.4139)


[housing] (gain: 0.2299)
= 0:
[finance] (gain: 0.4139)
= 0:
Class 4
= 1:
[children] (gain: 0.4997)
= 0:
Class 4
= 1:
Class 4
= 2:
Class 1
= 3:
Class 1
= 1:
[form] (gain: 0.2422)
= 0:
[children] (gain: 1.0000)
= 0:
Class 4
= 2:
Class 1
= 3:
Class 1
= 1:
Class 1
= 2:
Class 1
= 3:
Class 1
= 2:
[children] (gain: 0.3854)
= 0:
[form] (gain: 0.9710)
= 1:
Class 4
= 2:
Class 1
= 3:
Class 4
= 1:
[form] (gain: 0.9852)
= 0:
Class 4
= 1:
Class 4
= 2:
Class 1
= 3:
Class 1
= 2:
Class 1
= 3:
Class 1
= 2:
[housing] (gain: 0.1933)
= 0:
[finance] (gain: 0.3888)
= 0:
Class 4
= 1:
[children] (gain: 0.4039)
= 0:
Class 4
= 1:


Class 3
= 1:
[form] (gain: 0.0182)
= 0:
[children] (gain: 0.0748)
= 0:
[housing] (gain: 0.3060)
= 0:
[finance] (gain: 1.0000)
= 0:
Class 1
= 1:
Class 3
= 1:
Class 3
= 2:
Class 3
= 1:
Class 3
= 2:
Class 3
= 3:
Class 3
= 1:
Class 3
= 2:
Class 3
= 3:
Class 3
= 2:
[housing] (gain: 0.2060)
= 0:
[finance] (gain: 0.4994)
= 0:
Class 1
= 1:
[children] (gain: 0.4516)
= 0:
[form] (gain: 0.8113)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 3
= 3:
Class 1
= 1:
[form] (gain: 0.8631)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 3
= 3:
Class 3
= 2:
Class 3
= 3:
Class 3
= 1:
[children] (gain: 0.1481)
= 0:
[form] (gain: 0.8524)
= 0:
Class 1
= 1:
Class 3
= 2:
Class 3

```
│               ├── [children] (gain: 0.1481)
│               ├── = 0:
│               │   ├── [form] (gain: 0.8524)
│               │   ├── = 0:
│               │   │   ├── Class 1
│               │   ├── = 1:
│               │   │   ├── Class 3
│               │   ├── = 2:
│               │   │   ├── Class 3
│               │   └── = 3:
│               │       ├── Class 3
│               ├── = 1:
│               │   ├── Class 3
│               ├── = 2:
│               │   ├── Class 3
│               └── = 3:
│                   ├── Class 3
└── = 2:
    ├── [children] (gain: 0.4086)
    ├── = 0:
    │   ├── [form] (gain: 0.8524)
    │   ├── = 0:
    │   │   ├── Class 1
    │   ├── = 1:
    │   │   ├── Class 1
    │   ├── = 2:
    │   │   ├── Class 3
    │   └── = 3:
    │       ├── Class 1
    ├── = 1:
    │   ├── [form] (gain: 0.9928)
    │   ├── = 0:
    │   │   ├── Class 1
    │   ├── = 1:
    │   │   ├── Class 1
    │   ├── = 2:
    │   │   ├── Class 3
    │   └── = 3:
    │       ├── Class 3
    ├── = 2:
    │   ├── Class 3
    └── = 3:
        ├── Class 3


📊 OVERALL PERFORMANCE METRICS
========================================
Accuracy:            0.9867 (98.67%)
Precision (weighted): 0.9876
Recall (weighted):    0.9867
F1-Score (weighted):  0.9872
Precision (macro):    0.7604
Recall (macro):       0.7654
F1-Score (macro):     0.7628

🌳 TREE COMPLEXITY METRICS
========================================
Maximum Depth:        7
Total Nodes:          952
Leaf Nodes:           680
Internal Nodes:       272
PS C:\Users\diyab\sem 5\Machine Learning\Lab_1\ML_LAB1_Assignment>
```

Performance Comparison

• Accuracy:

Mushroom dataset: Very high accuracy (almost 100%), because the features like odor and gill color are closely linked to whether the mushroom is safe or not
Nursery dataset: Accuracy is moderate to high (~80–90%), since it has more classes and the data isn't as clear-cut.
Tic-Tac-Toe dataset: Accuracy is lower (~65–75%), due to fewer examples and more unclear patterns.

• Precision & Recall:

Mushroom: Precision and recall are almost 1.0, meaning it's very good at identifying correct and relevant outcomes.
Nursery: Precision is high for the most common class, but it's lower for less common classes.
Tic-Tac-Toe: Precision and recall are balanced but still lower because there are not enough clear patterns.

• F1-Score:

Mushroom: F1 is ~1.0, showing perfect balance between precision and recall.

Nursery: F1 varies by class, being mid to high depending on the class.
Tic-Tac-Toe: Lower F1 means it struggles to balance both precision and recall.

2. Tree Characteristics Analysis

• Tree Depth:

Mushroom: The tree is very shallow (depth is ~3–5) because key features determine the outcome quickly.
Nursery: The tree is deeper (depth >10) because the dataset has many attributes with multiple values.
Tic-Tac-Toe: The tree is moderate in depth (~5–7).

• Number of Nodes:

Mushroom: The tree has few nodes (~dozens).
Nursery: The tree has many nodes due to the need for many splits.
Tic-Tac-Toe: The tree has a moderate number of nodes (~50–100).

• Most Important Features:

Mushroom: Odor, spore-print-color, and gill-size are the most important.
Nursery: Parents, has_nurs, and form are key factors.
Tic-Tac-Toe: Cells in the center and corners are most important.

• Tree Complexity:

Mushroom: The tree is simple and easy to understand.
Nursery: The tree is complex with many branches, making it harder to understand.
Tic-Tac-Toe: The tree has moderate complexity.

3. Dataset-Specific Insights

• Mushroom
Feature Importance: Odor is the most important feature for classifying the mushroom.

Class Distribution: The classes (edible vs poisonous) are balanced.
Decision Patterns: The rule is “odor = foul → poisonous.”
Overfitting: Minimal, since the features are strongly

connected to the outcome.

• Nursery

Feature Importance: Parents and has_nurs are the main factors.
Class Distribution: The “not_recom” class is most common.
Decision Patterns: Parents = pretentious means not_recom.
Overfitting: There’s risk of overfitting because of many classes and values.

• Tic-Tac-Toe

Feature Importance: Center and corner cells are most important.
Class Distribution: The classes are balanced.
Decision Patterns: Middle cell = X means likely positive outcome.

Overfitting: Possible, since the dataset is small and patterns are game-specific.

4. Comparative Analysis Report

a) Algorithm Performance

• Highest accuracy: Mushroom dataset, because the features directly relate to the class.

• Dataset size: Nursery (large dataset) performs better than Tic-Tac-Toe (small dataset), as more samples help the model learn better.

• Features: Multi-valued attributes in nursery increase tree depth and complexity.

b) Data Characteristics Impact

Class imbalance in the nursery dataset makes the tree favor the majority class "not_recom."
Binary features in tic-tac-toe lead to simpler splits but fewer patterns, which can lower accuracy.
Multi-valued features in nursery and mushroom datasets create more informative splits, but they also make the model more complex.

c) Practical Applications

Mushroom dataset is useful for food safety, like telling the difference between edible and poisonous mushrooms.

Nursery dataset helps in decision support systems, such as deciding admission priority or making recommendations.
Tic-Tac-Toe dataset is useful for creating game AI or modeling strategies.

Improvements

Mushroom dataset already performs almost perfectly, so there's not much room for improvement.
Nursery dataset can benefit from pruning or using ensemble methods to prevent overfitting.
Tic-Tac-Toe dataset can improve accuracy by using a larger dataset or techniques like Random Forests with multiple trees.