

# ***ML LAB \_3***

**Student Name:** Mohammed Bilal

**SRN:** PES2UG23CS344

**Course:** Machine Learning Lab

**Experiment:** Lab 3 – Implementation of ID3 Decision Tree

---

## **1. Introduction**

The objective of this lab is to implement the ID3 Decision Tree algorithm and analyze its performance across three datasets: Mushroom Classification, Tic-Tac-Toe Endgame, and Nursery School Admissions. The results were evaluated using accuracy, precision, recall, F1-score, and tree complexity metrics. The aim is to understand how dataset characteristics affect decision tree performance.

---

## **2. Observations**

### ***Mushroom Dataset***

- **Accuracy:** 100%
- **Precision/Recall/F1:** 1.0 for all metrics.
- **Tree Depth:** 4
- **Total Nodes:** 29 (24 leaves, 5 internal).
- **Remarks:**
  - Perfect classification due to highly discriminative features like *odor* and *sporeprint color*.
  - Very small and interpretable tree.
  - No signs of overfitting.

## Performance Matrix (Mushroom Dataset):

```
■ 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
```

```
PS C:\Users\Shreyas\Downloads\ML_LAB_1> python test.py --lr 0.01 --n_epochs 1000 --model mlp --dataset mushroom --print--loss
Running tests with PYTORCH framework
=====
target column: 'class' (last column)
dataset info:
shape: (4204, 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-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spore-print-color', 'population', 'habitat', 'class')
First few rows:
cap-shape: ['g', 'f', 'n', 'b', 's'] -> [0, 0, 0, 0, 0]
cap-surface: ['s', 'y', 'f', 'g'] -> [2, 3, 0, 1]
cap-color: ['n', 'y', 'w', 'g'] -> [4, 9, 0, 2]
class: ['e', 'w'] -> [0, 0]
Processed dataset shape: torch.Size([4204, 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-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spore-print-color', 'population', 'habitat']
Target class: 'class'
Framework: PYTORCH
Data type: <class 'torch.Tensor'>
=====
DECISION TREE CONSTRUCTION LOGO
=====
Total samples: 4204
Training samples: 3680
Testing samples: 524
Constructing decision tree using Training data...
④ Decision tree construction completed using PYTORCH!
⑤ DECISION TREE STRUCTURE
=====
Decision Tree Structure:
Root [Layer 0] (split: 0.4800)
├── 0
│   ├── Class 4
│   ├── 2 1:
│   │   ├── Class 1
│   │   ├── 1 2:
│   │       ├── Class 1
│   │       ├── 4 3:
│   │           ├── Class 0
│   │           └── Class 1
```

```
    └── 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
```

```
  = 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
```

---

## Tic-Tac-Toe Dataset

- Accuracy: 87.30%

- **Precision/Recall/F1 (weighted)**: ~0.87
- **Precision/Recall/F1 (macro)**: ~0.86
- **Tree Depth**: 7
- **Total Nodes**: 281 (180 leaves, 101 internal).
- **Remarks**:
  - The dataset is complex because many board states look similar but have different outcomes.
  - Accuracy is moderate and the tree is deeper, suggesting difficulty in generalizing rules.
  - Possible overfitting as the tree grows large without achieving very high accuracy.

### Performance Matrix (Tictactoe Dataset):

OVERALL PERFORMANCE METRICS	
Accuracy:	0.8730 (87.30%)
Precision (weighted):	0.8741
Recall (weighted):	0.8730
F1-Score (weighted):	0.8734
Precision (macro):	0.8590
Recall (macro):	0.8638
F1-Score (macro):	0.8613
TREE COMPLEXITY METRICS	
Maximum Depth:	7
Total Nodes:	281
Leaf Nodes:	180
Internal Nodes:	101

---

### Nursery Dataset

- **Accuracy**: 98.67%
- **Precision/Recall/F1 (weighted)**: ~0.98
- **Precision/Recall/F1 (macro)**: ~0.76

- **Tree Depth:** 7
- **Total Nodes:** 952 (680 leaves, 272 internal).
- **Remarks:**
  - Excellent weighted accuracy, but macro values are lower due to class imbalance.
  - The tree is very large, reflecting the dataset's size and multi-class nature.

### Performance Matrix (Nursery Dataset):

```

 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\Shreyas\Downloads\ML_LAB_1>

```

## 3. Comparative Analysis

Dataset	Accurac y	Tree Depth	Total Nodes	Remarks
Mushroom	100%	4	29	Perfect, small tree, clean data
Tic-Tac-To	87.3%	7	281	Ambiguous states, moderate accuracy
Nursery			98.67%	7      952
				Very high accuracy, imbalance affects minority classes
•				

**Best Performing Dataset:** Mushroom → because of highly discriminative features and balanced data.

- **Most Complex Tree:** Nursery → large number of attributes and classes increases tree size.
  - **Lowest Accuracy:** Tic-Tac-Toe → due to complexity of board game states and overlapping decision patterns.
- 

## **4. Insights and Discussion**

- **Effect of Data Size:** Larger datasets like Nursery give high weighted accuracy but may overfit.
- **Effect of Features:** Highly discriminative features (Mushroom) allow shallow trees with perfect accuracy.
- **Effect of Class Imbalance:** Nursery dataset shows imbalance reduces macro precision/recall.
- **Overfitting:** Both Tic-Tac-Toe and Nursery datasets show deeper and larger trees, which may not generalize well.

### **Decision Trees:**

#### **1) Mushrooms Dataset:**

▲ DECISION TREE STRUCTURE

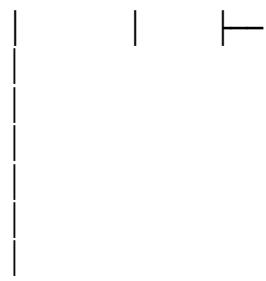
---

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: |   |   |   |   |   Class 0 |   |
    |   |   |   | = 2:
    |   |   |   |   | [cap-color] (gain: 0.7300)
    |   |   |   |   | = 1: |   |   |   |   Class
    |   |   |   |   | = 4: |   |   |   |   |   |
    |   |   |   |   | Class 0 |   |   |   | = 8: |   |
    |   |   |   |   | Class 1 |   |   |   |   | = 9: |
    |   |   |   |   |   | Class 1 |   |   |
    |   |   |   |   | = 4: |   |   |   |
    |   |   |   | Class 0 |   |   | = 6: |
    |   |   |   | Class 0 |   |   |   | = 8: |
    |   |   |   | Class 0 |   |   | = 6: |
    |   |   |   | Class 1 |   |   | = 7: |   |   Class
    |   |   |   | = 8:
    |   |   |   | Class 1 |

```



## 2) Tictactoe Dataset:

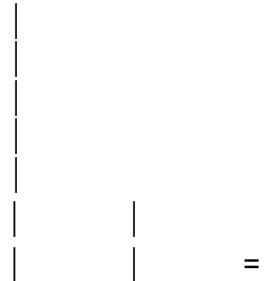
### DECISION TREE STRUCTURE

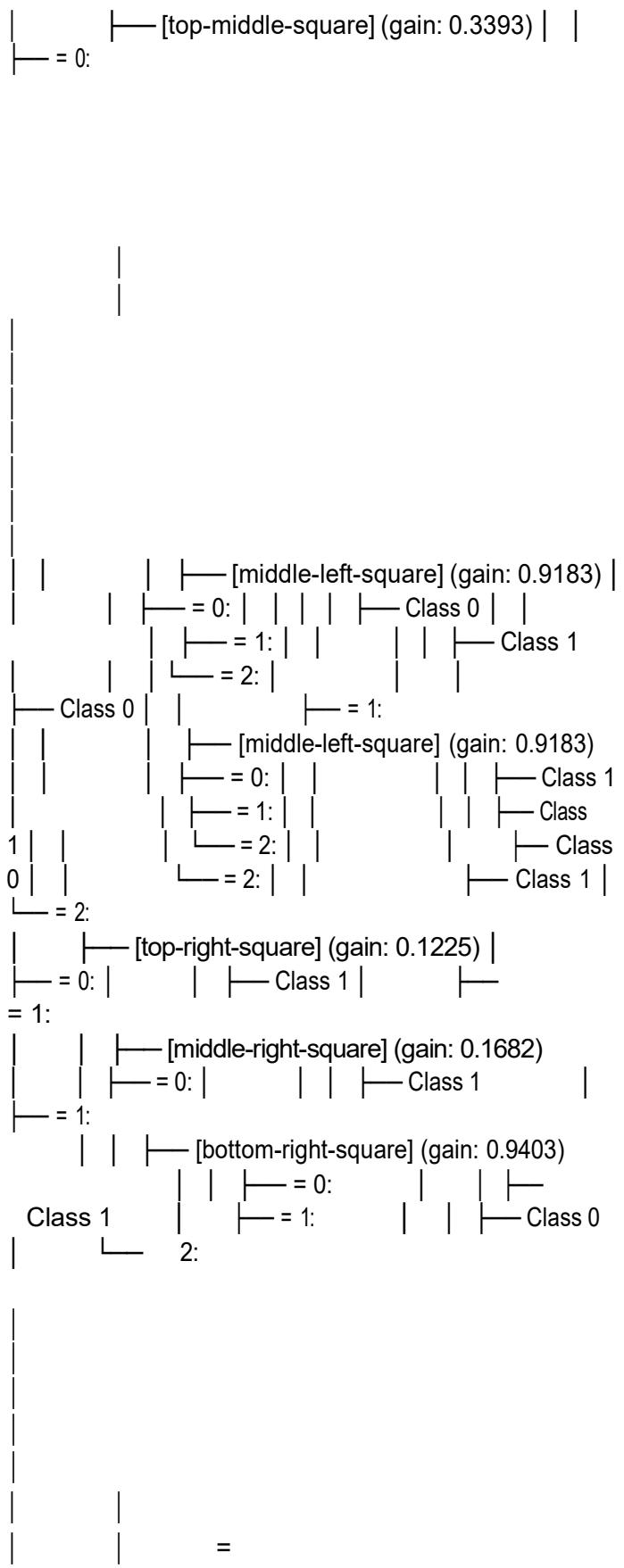
=

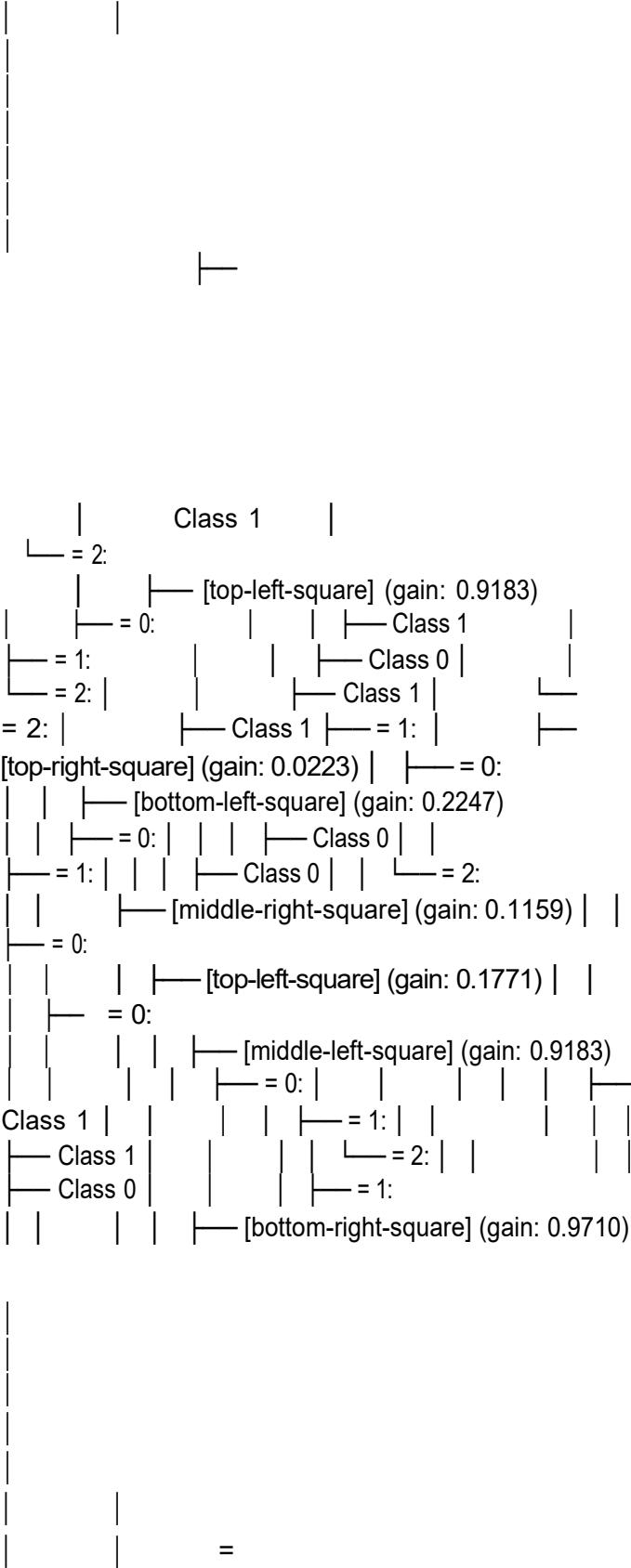
```

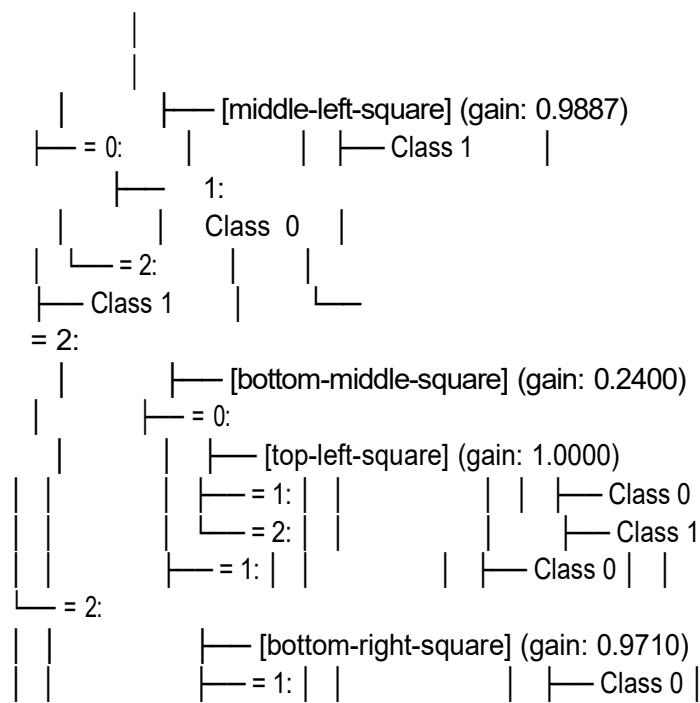
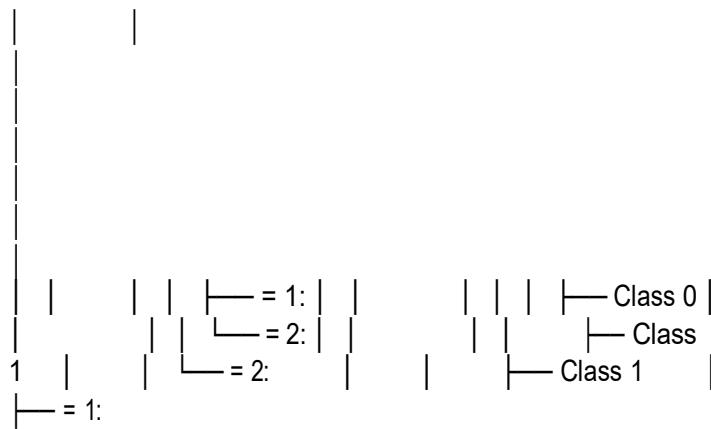
Root [middle-middle-square] (gain: 0.0834) ┌── = 0:
|   └── [bottom-left-square] (gain: 0.1056) |   ┌──
= 0:
|   |   └── [top-right-square] (gain: 0.9024)
|   |   └── = 1: |   |   └── Class 0 |   |
|   |   └── = 2: |   |   └── Class 1 |   └── = 1:
|   |   └── [top-right-square] (gain: 0.2782)
|   |   └── = 0: |   |   └── Class 0 |   |
|   |   └── = 1: |   |   └── Class 0 |   └──
= 2:
|   |   └── [top-left-square] (gain: 0.1767) |   |
|   = 0:
|   |   └── [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: |   |   └── = 2: |   |

```

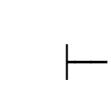




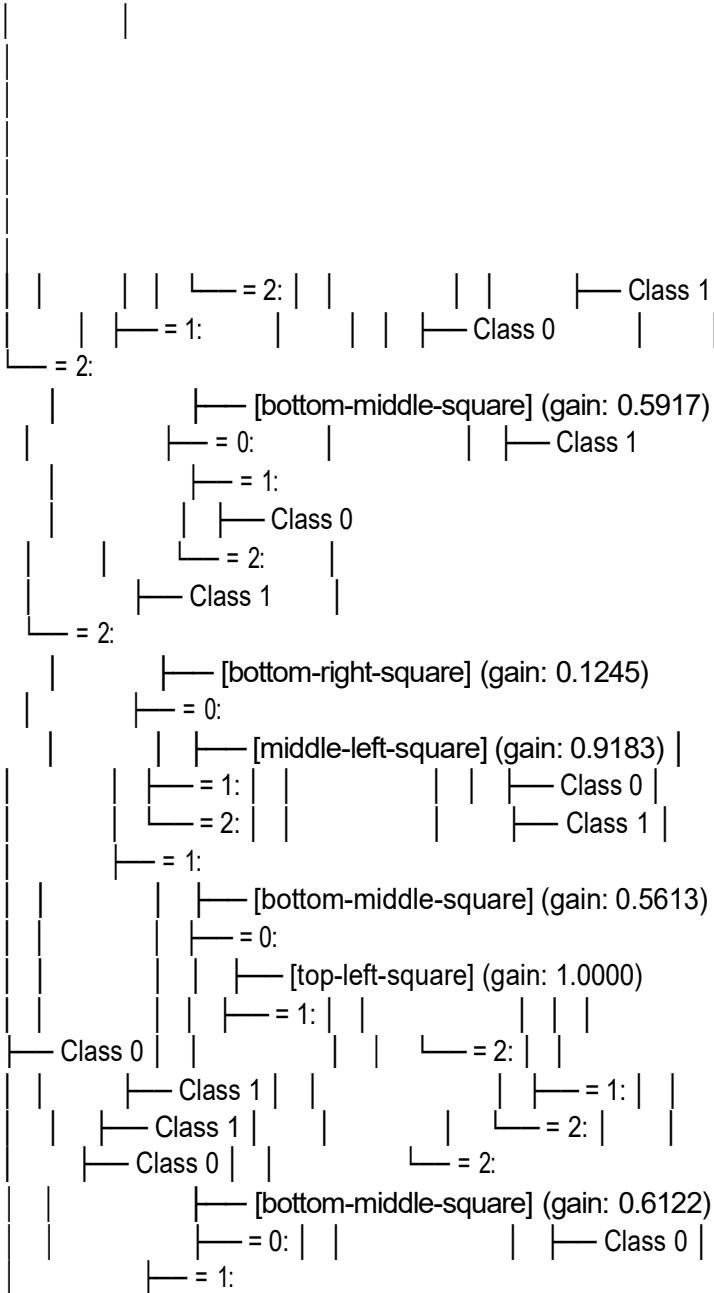


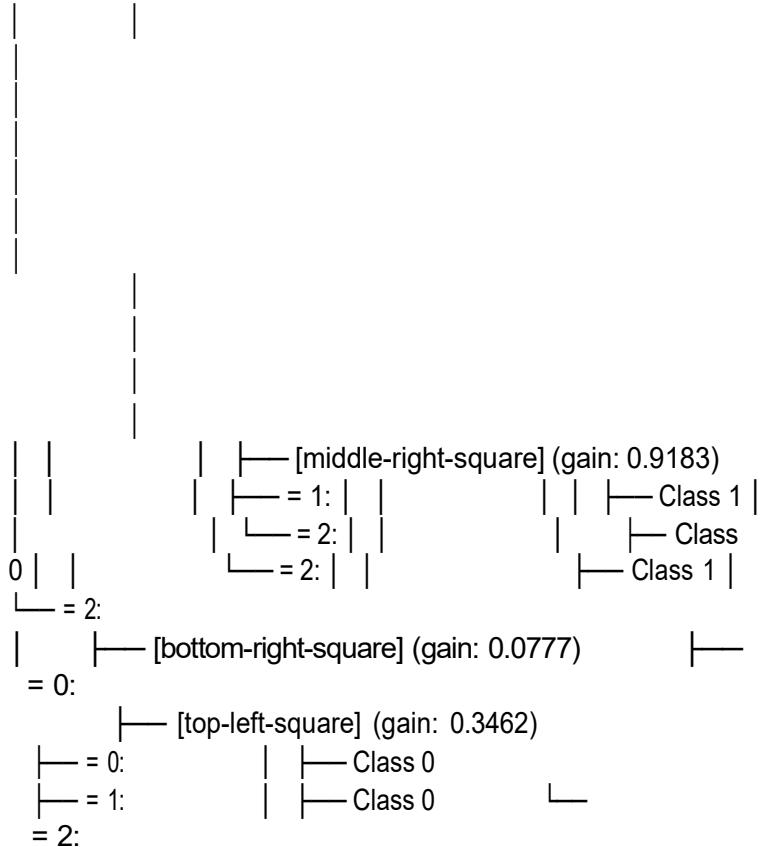


└ = 2: | | Class 1 |  
| = 1:  
| | [bottom-left-square] (gain: 0.4759)  
| | | = 0: | | Class 0 | |  
| = 1: | | | Class 0 | | └ = 2:  
| | [top-middle-square] (gain: 0.1974) |  
| | | = 0: | | Class 1 | |  
| = 1:  
| | [top-left-square] (gain: 0.3436) | |  
| | | = 0:  
| | | | [bottom-middle-square] (gain:  
0.9183)  
| | | | | = 1: | | Class 0



|

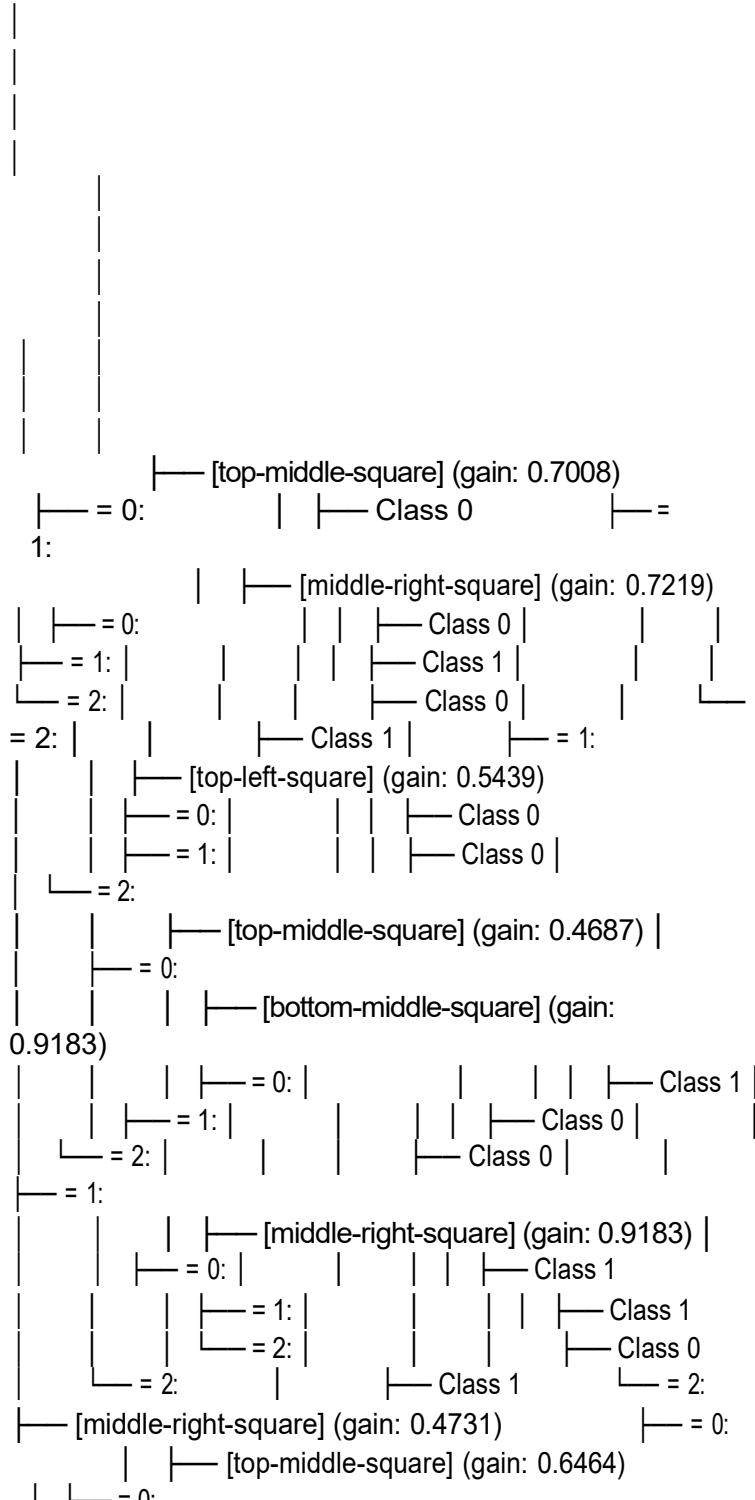


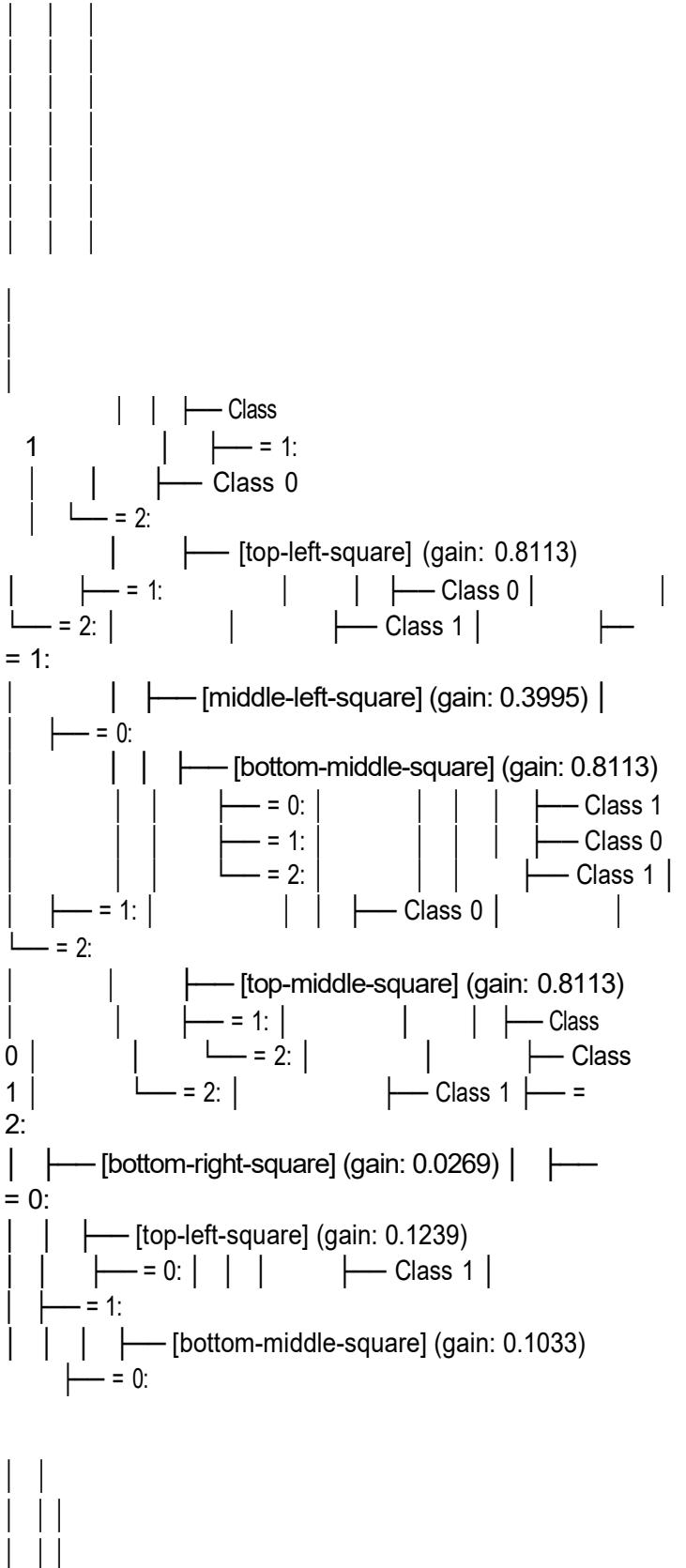


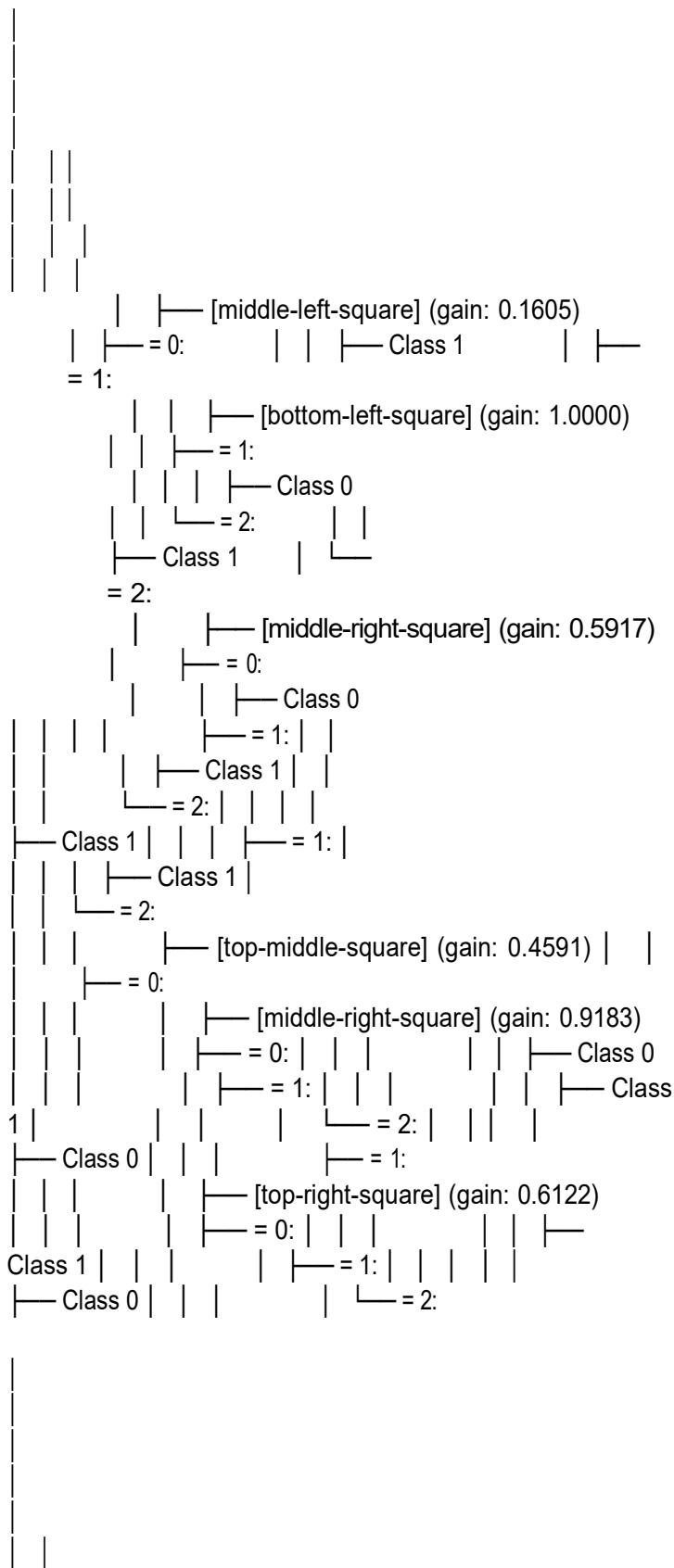
| |

|

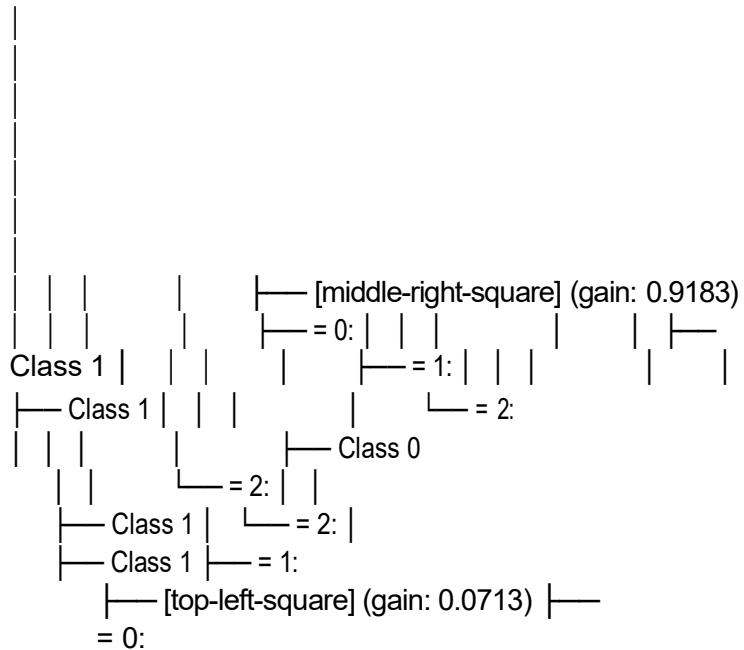
|

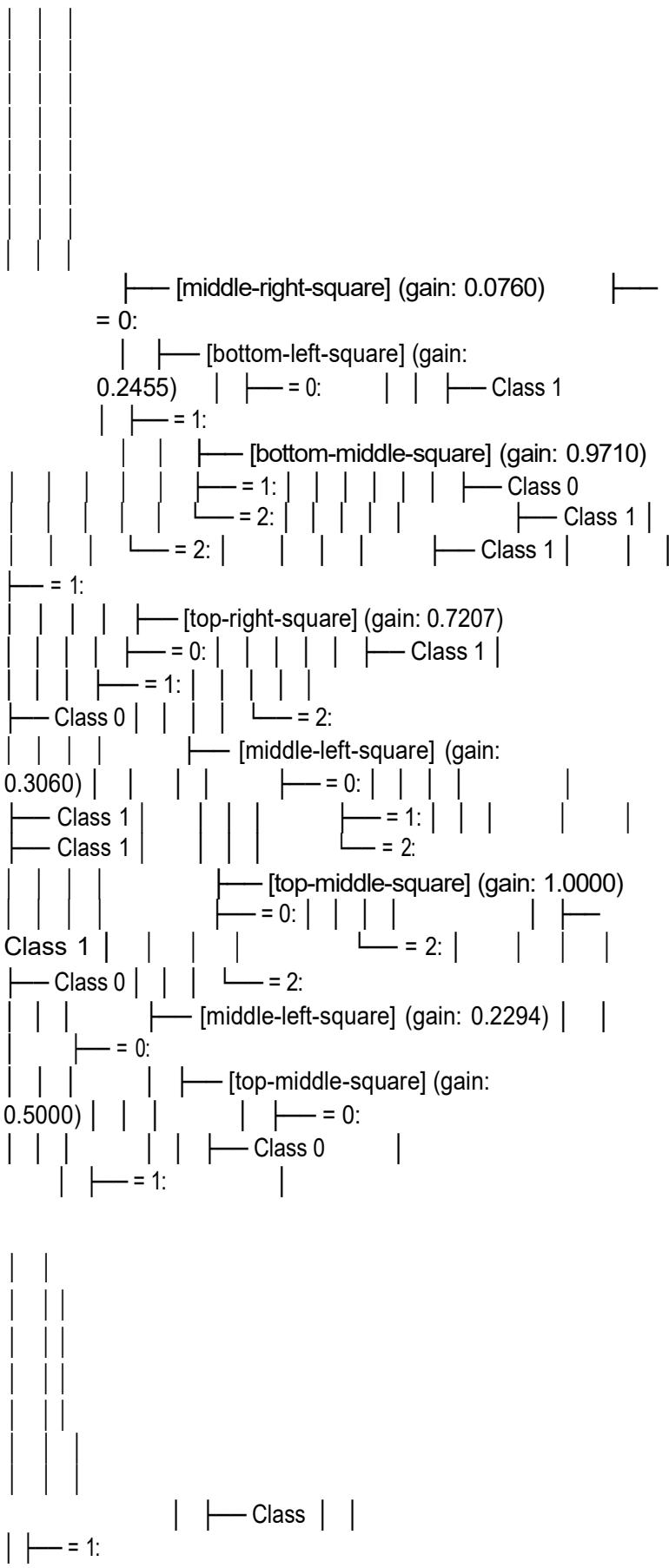






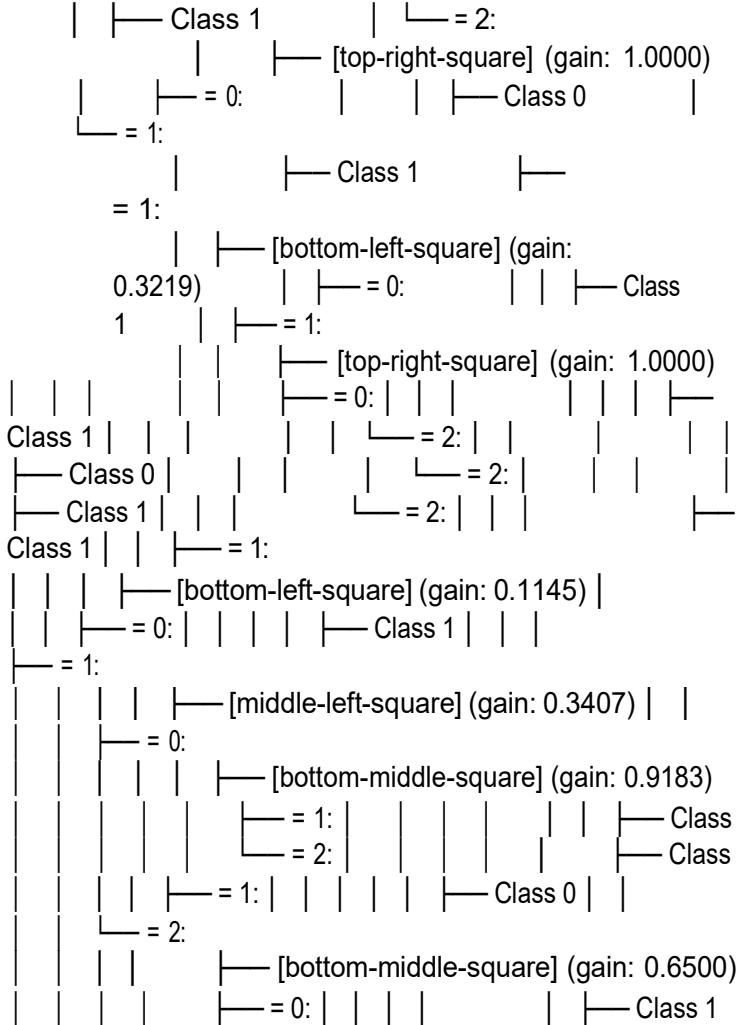






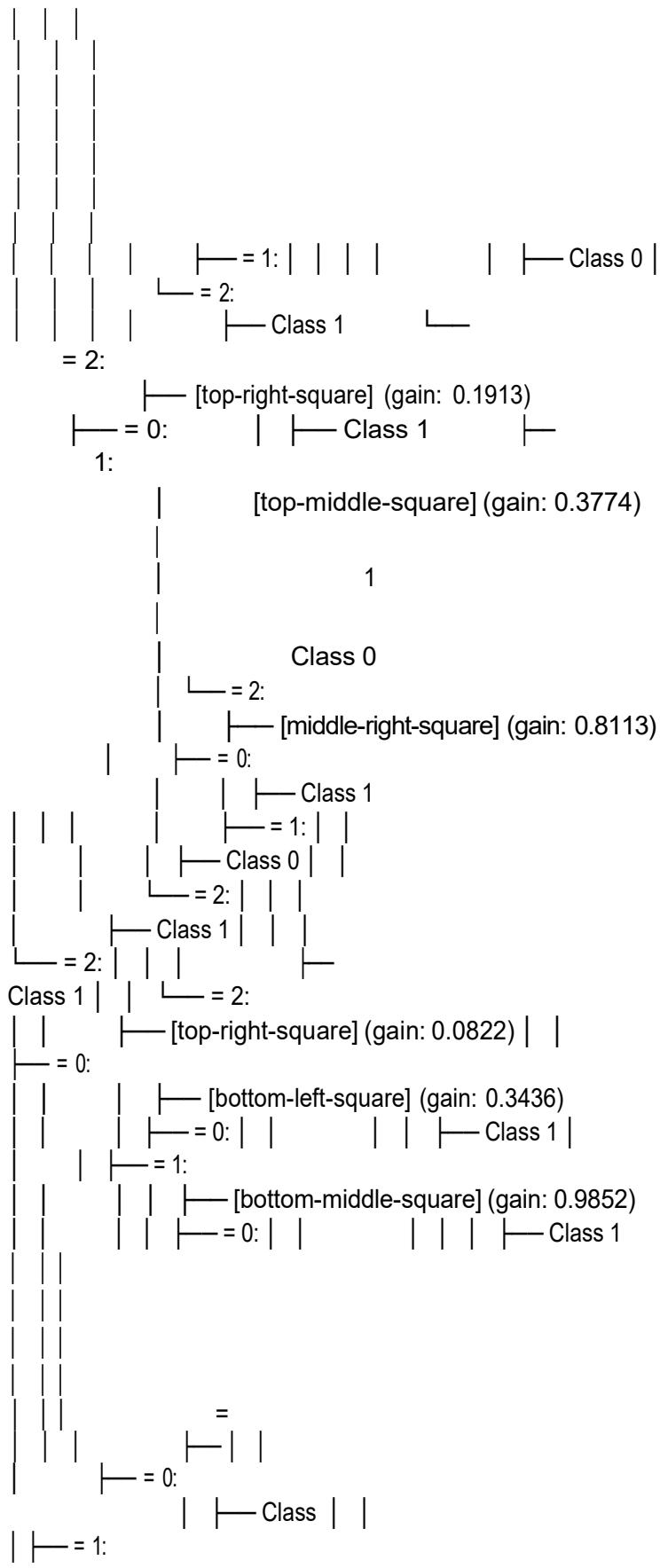
| | |

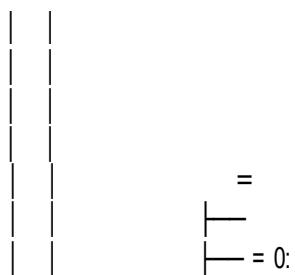
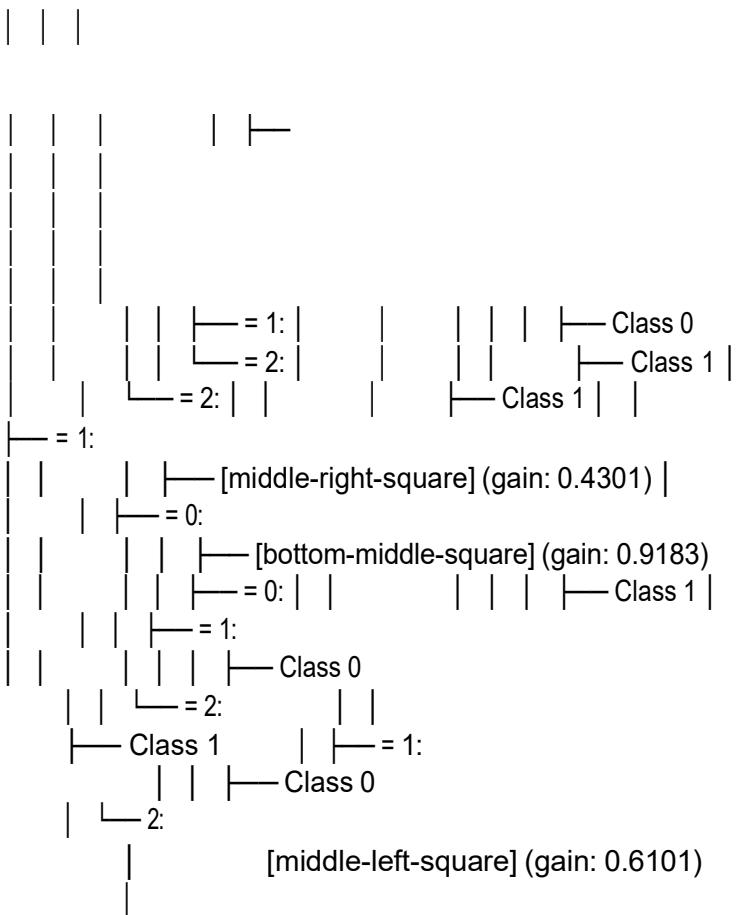
| | | | |

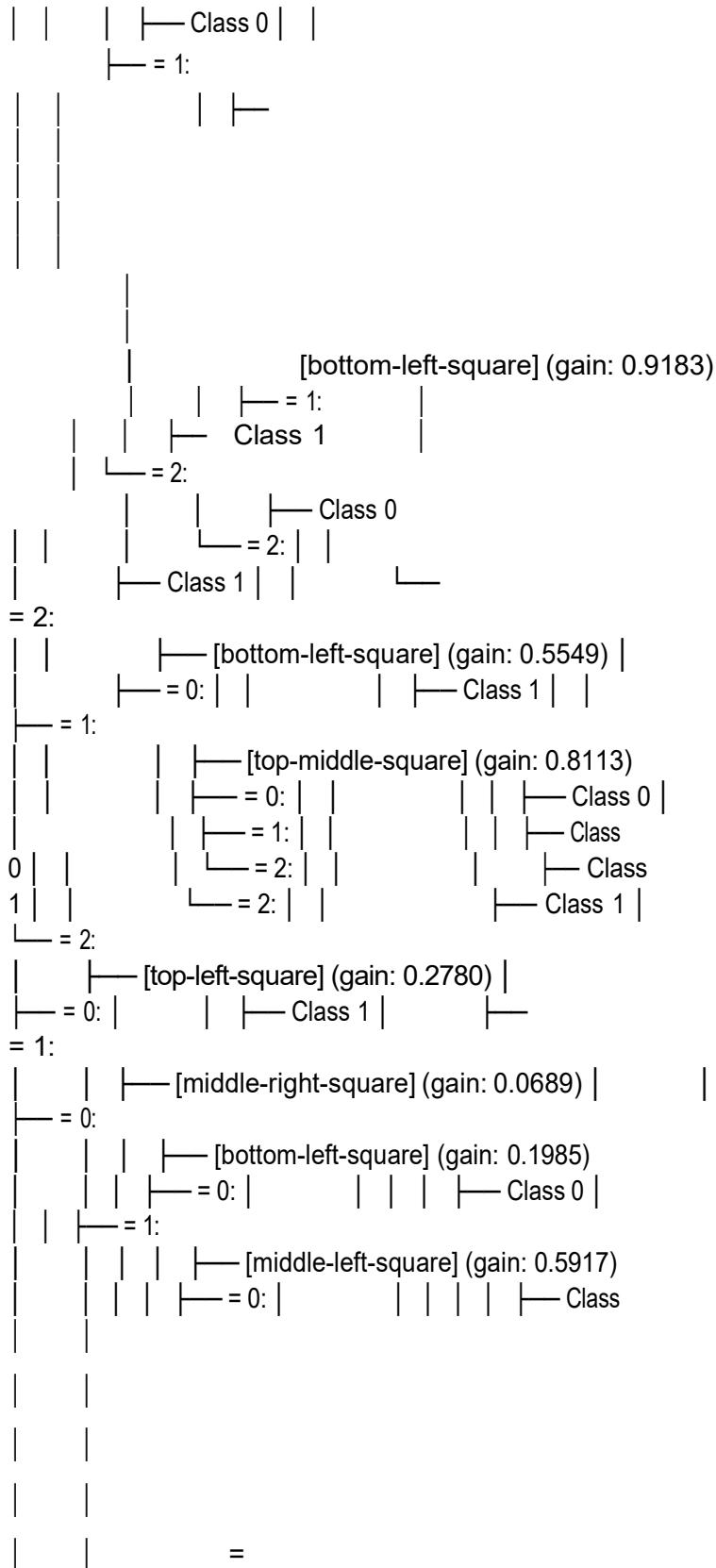


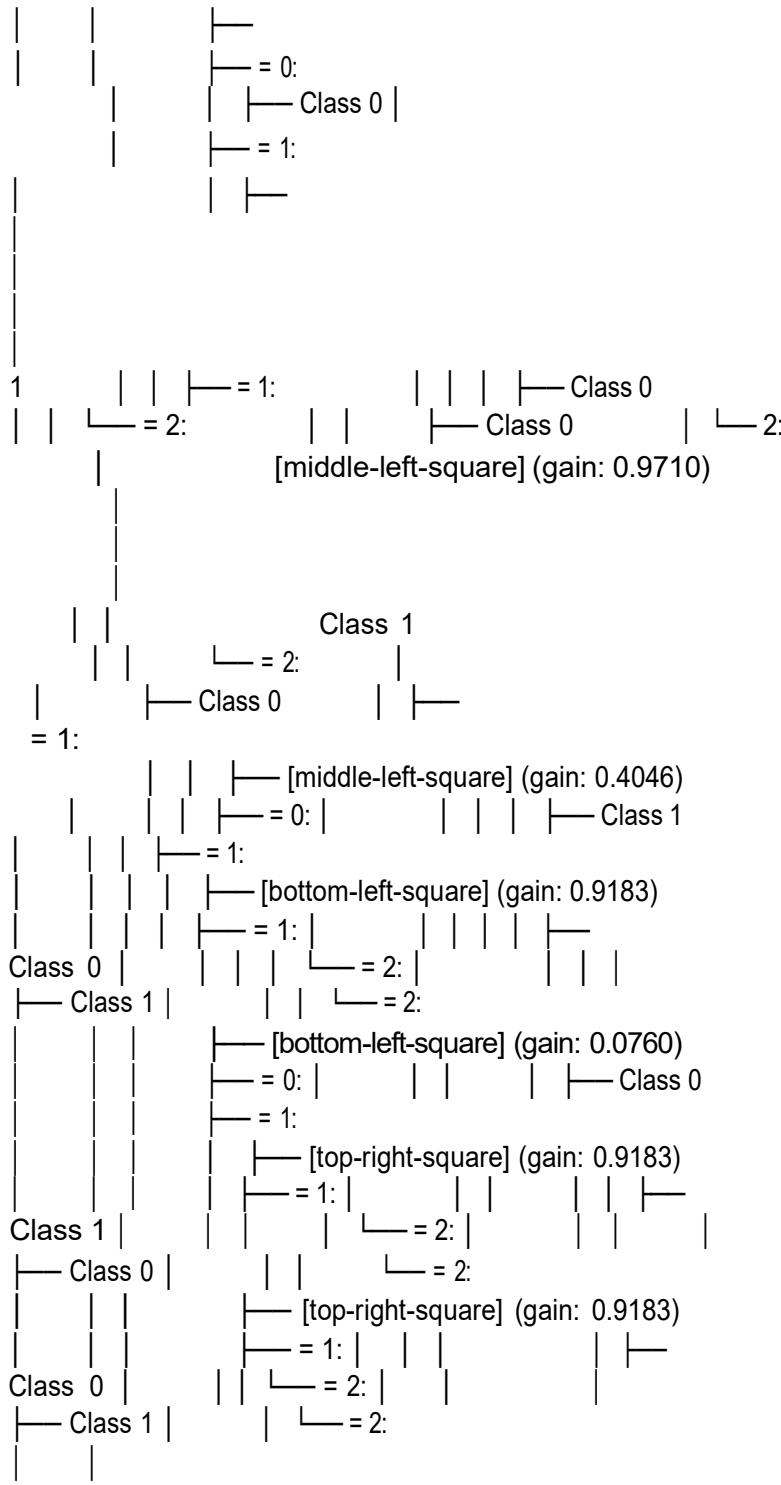
| | |

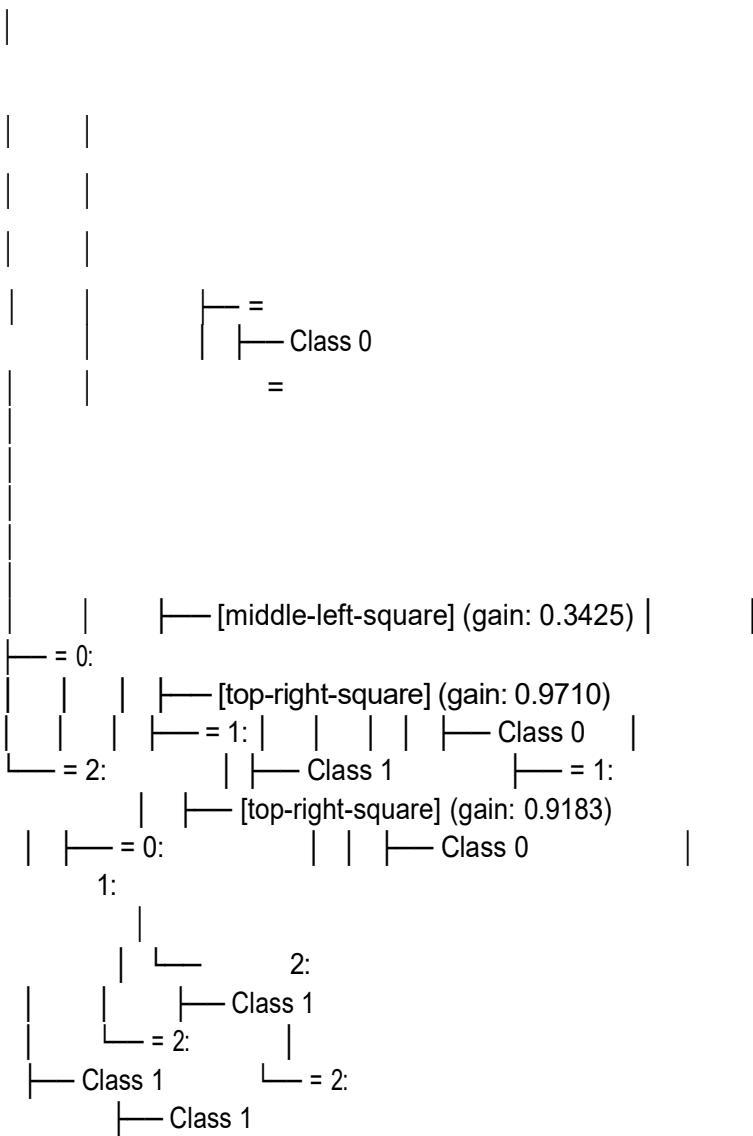
=  
| = 0:











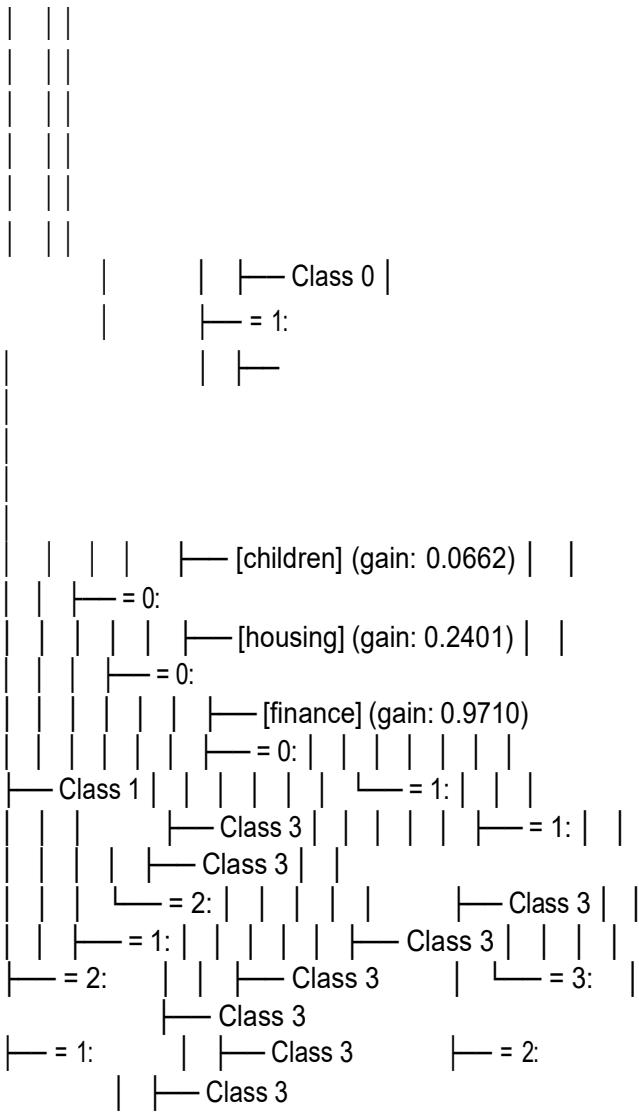
### 3) Nursery Dataset:

#### ▲ DECISION TREE STRUCTURE

```

=====
=
Root [health] (gain: 0.9595)
├── = 0: └── Class 0
├── = 1:
│   ├── [has_nurs] (gain: 0.3555)
│   │   ├── = 0:
│   │   │   ├── [parents] (gain: 0.1673)
│   │   │   │   ├── = 0:
│   │   │   │   │   ├── [form] (gain: 0.0171)
│   │   │   │   │   └── = 0:
|   |   |   |   |
|   |   |   |   |   └── Class 1
|   |   |   |
|   |   |   └── Class 1
|   |   |
|   |   └── Class 1
|   |
|   └── Class 1

```



|

| |

| |

| |

| |

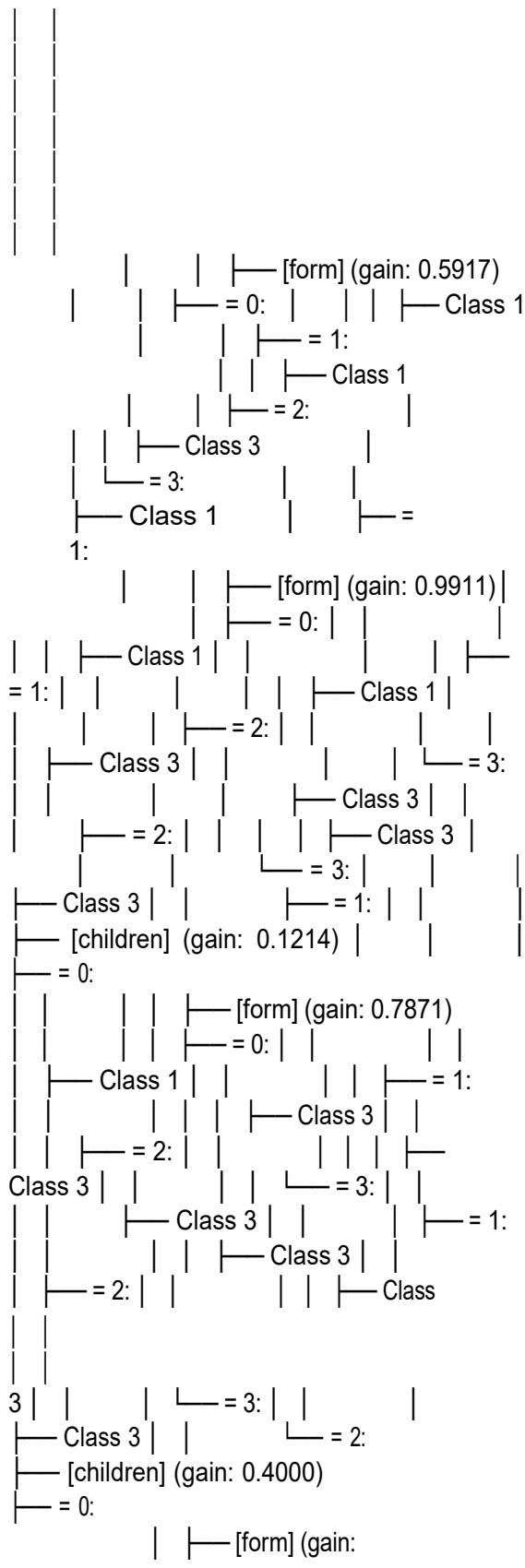
| |

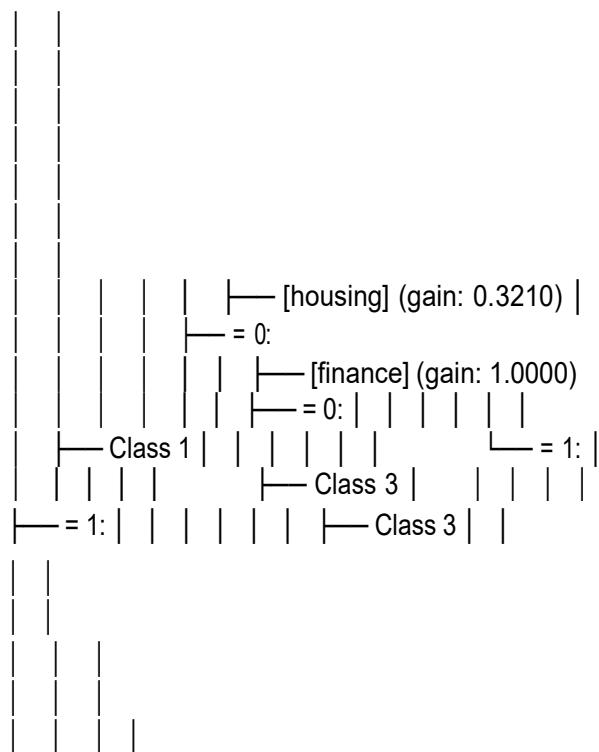
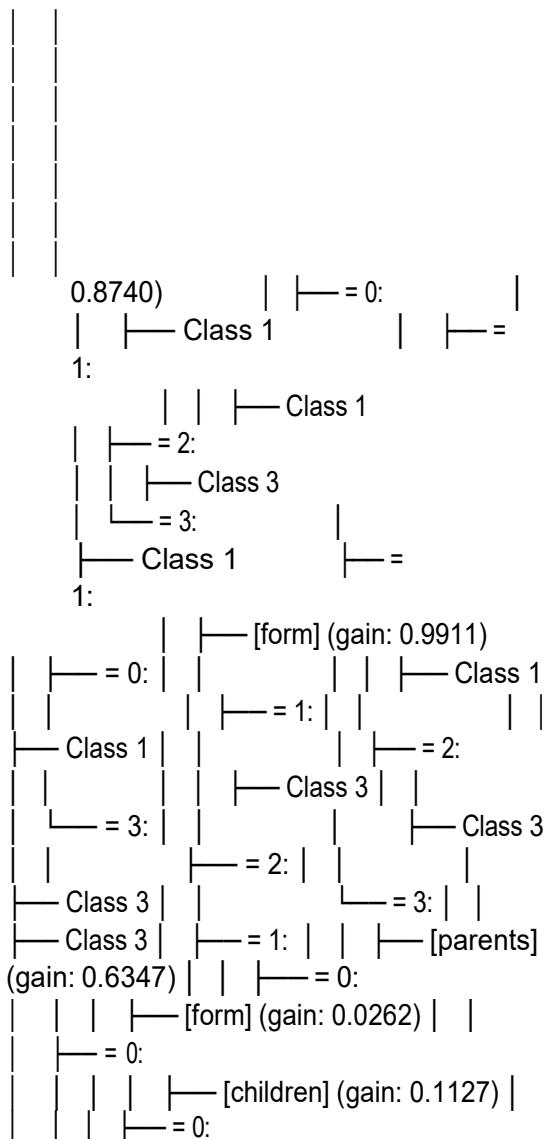
| |      |- =

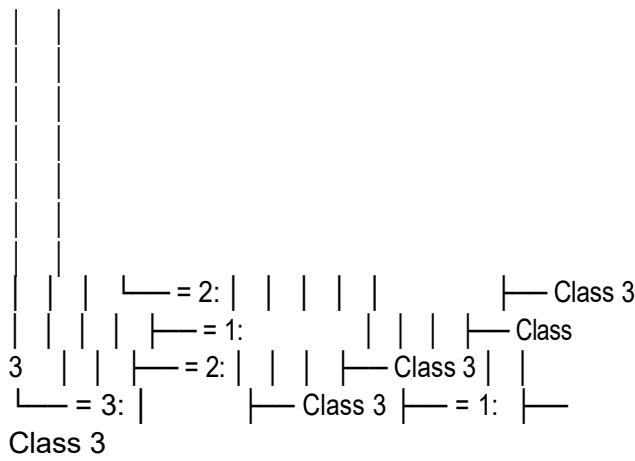
```

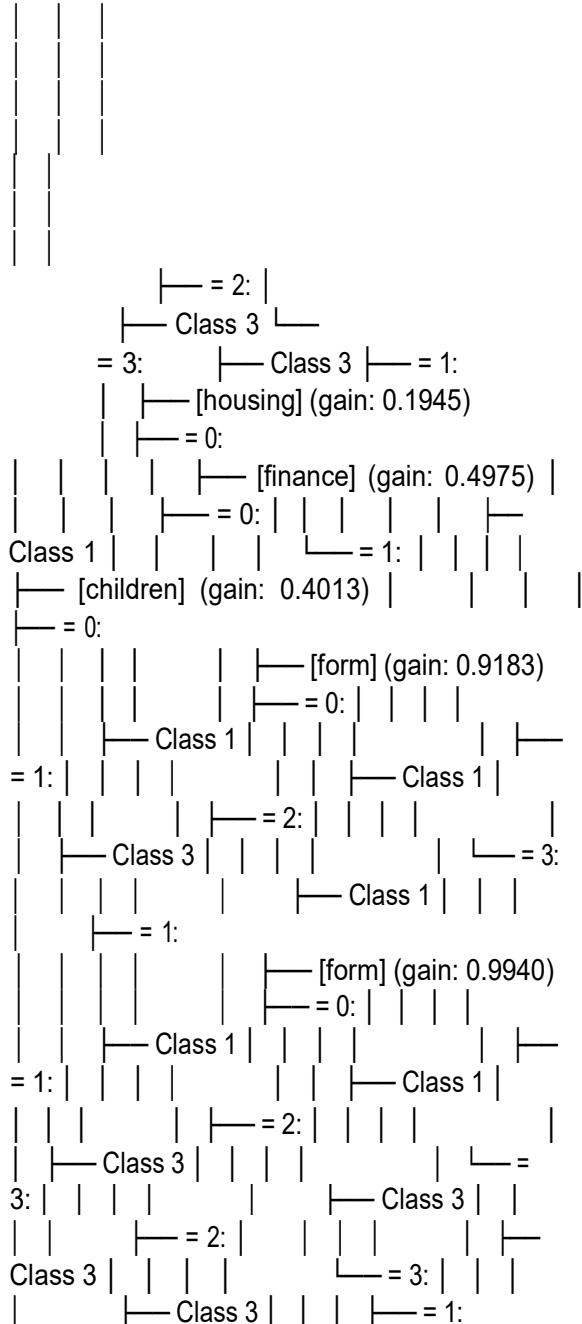
    └─ = 3:
      └─ Class 3 ──
      = 1:
        | └─ [form] (gain: 0.0269)
        | └─ = 0:
          |   └─ [children] (gain: 0.1080) ──
          |   └─ = 0:
            |     └─ [housing] (gain: 0.3219) ──
            |     └─ = 0:
              |       └─ [finance] (gain: 1.0000)
              |       └─ = 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: ┌─ ┌─ ┌─
          └─ = 2: ┌─ ┌─ └─ [housing] (gain: 0.2558) ┌─ ┌─
          └─ = 0:
            └─ ┌─ ┌─ └─ [finance] (gain: 0.4787)
            └─ Class 1 ┌─ ┌─ ┌─ ┌─ ┌─ ┌─ └─ = 0: ┌─ ┌─ ┌─
              └─ [children] (gain:
                0.5240) ┌─ ┌─ └─ = 0:

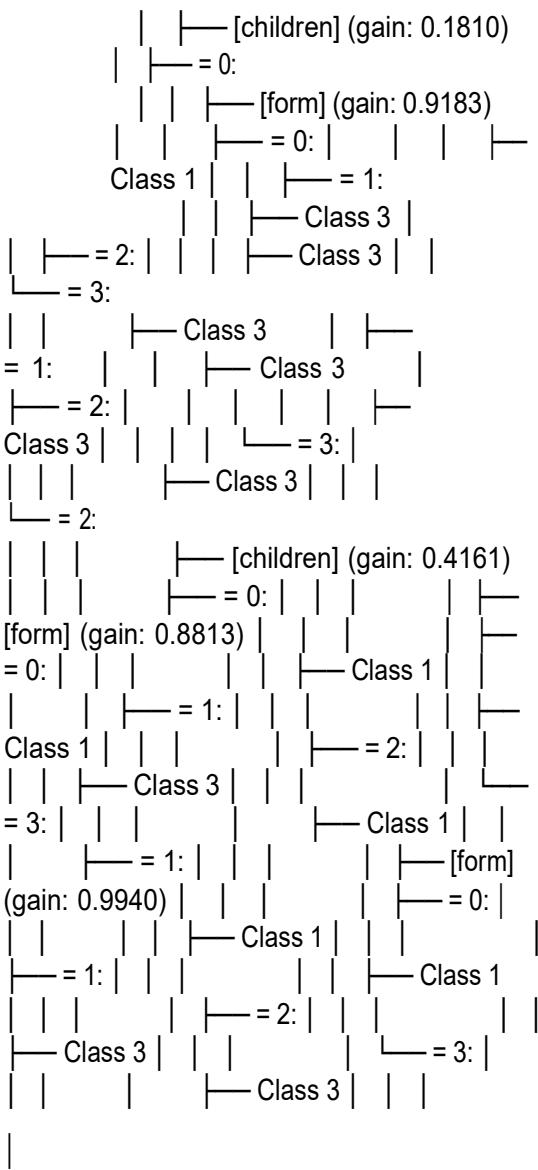
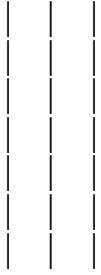
```

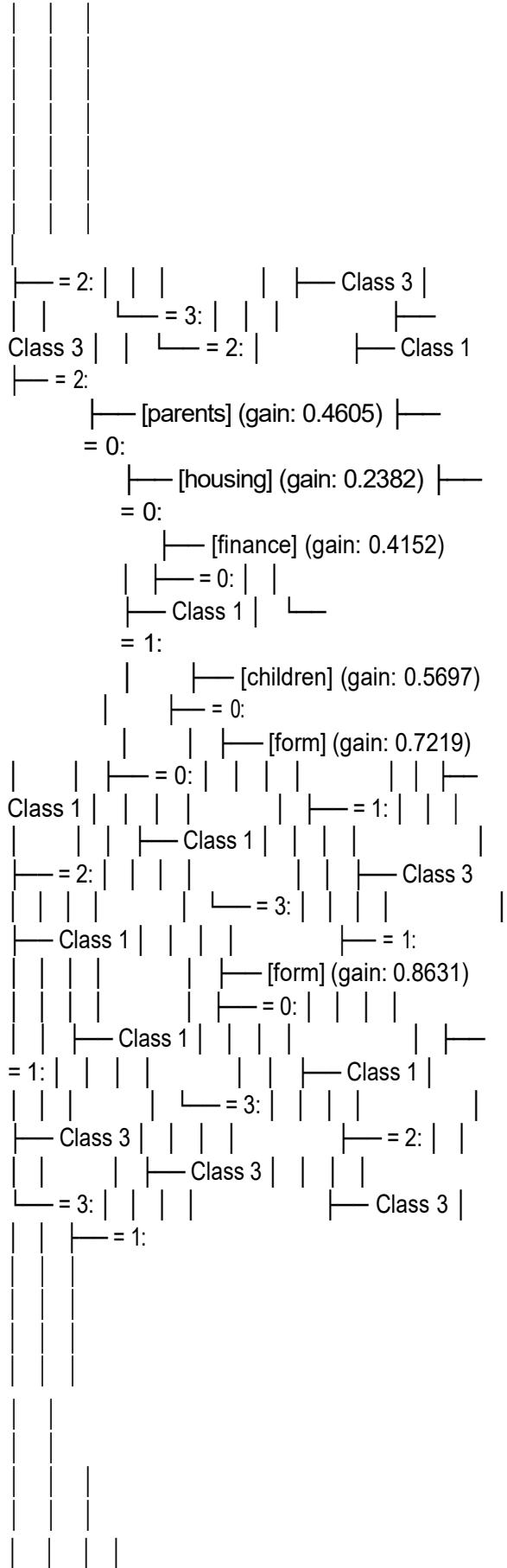


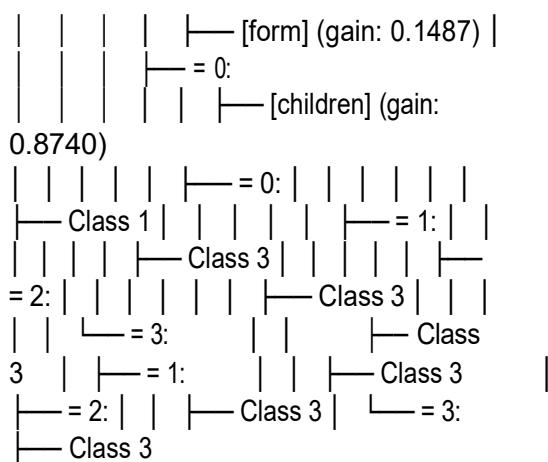
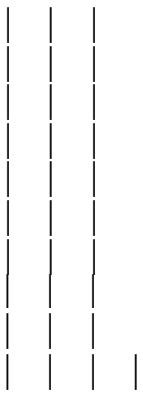


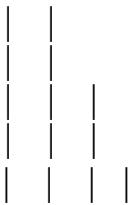
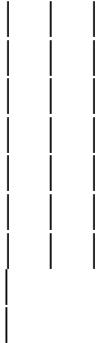






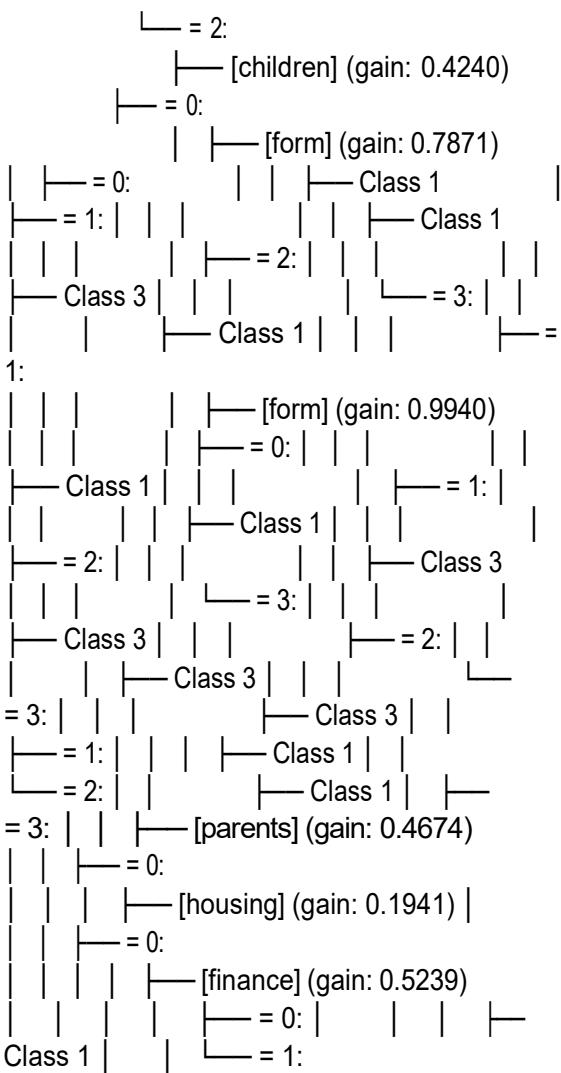






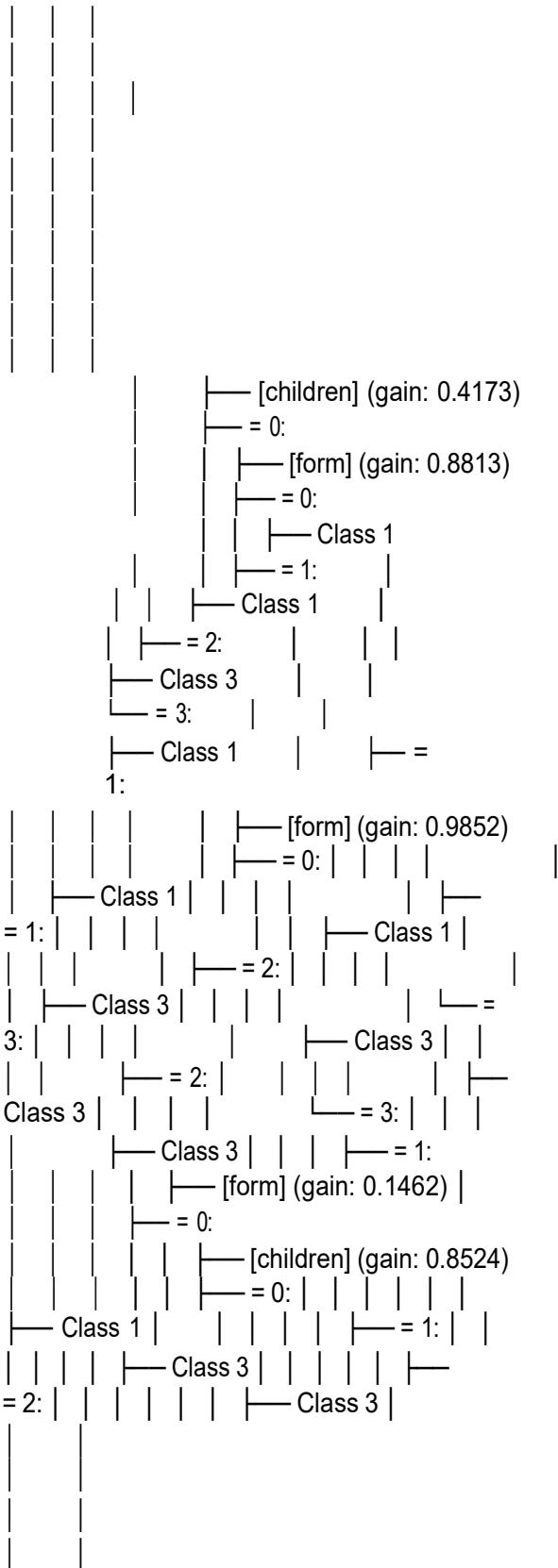
| | |

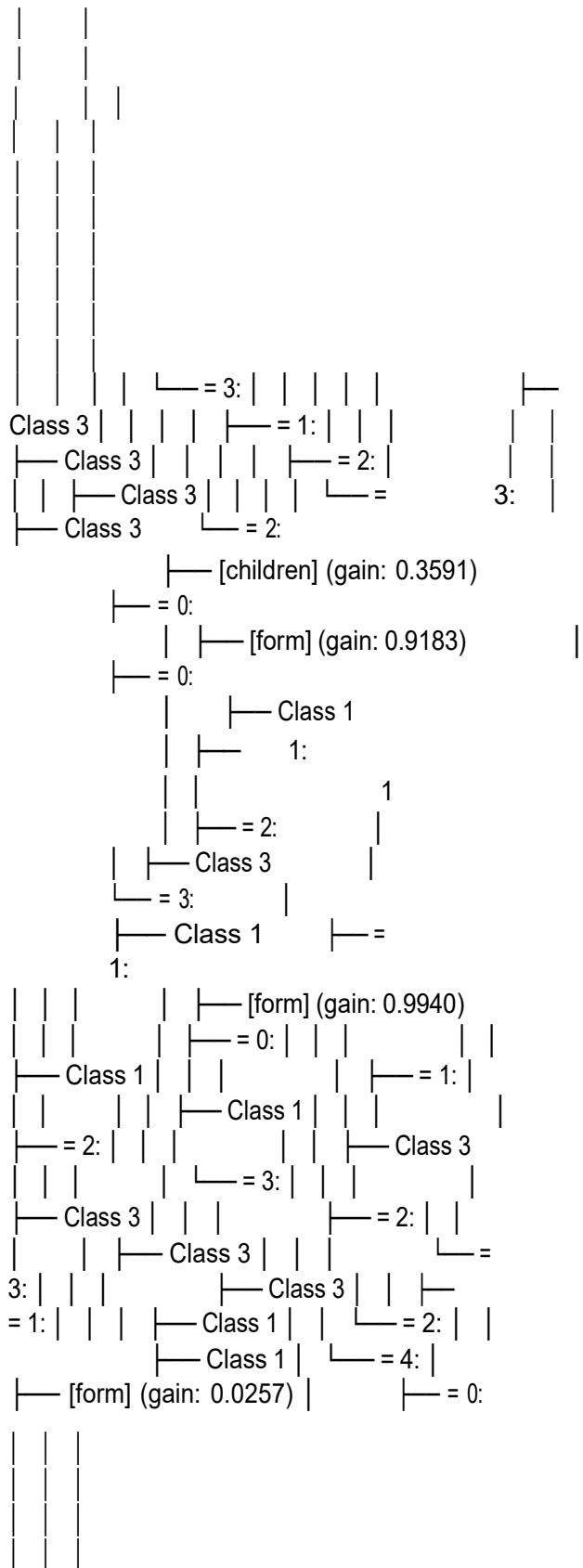
| | |



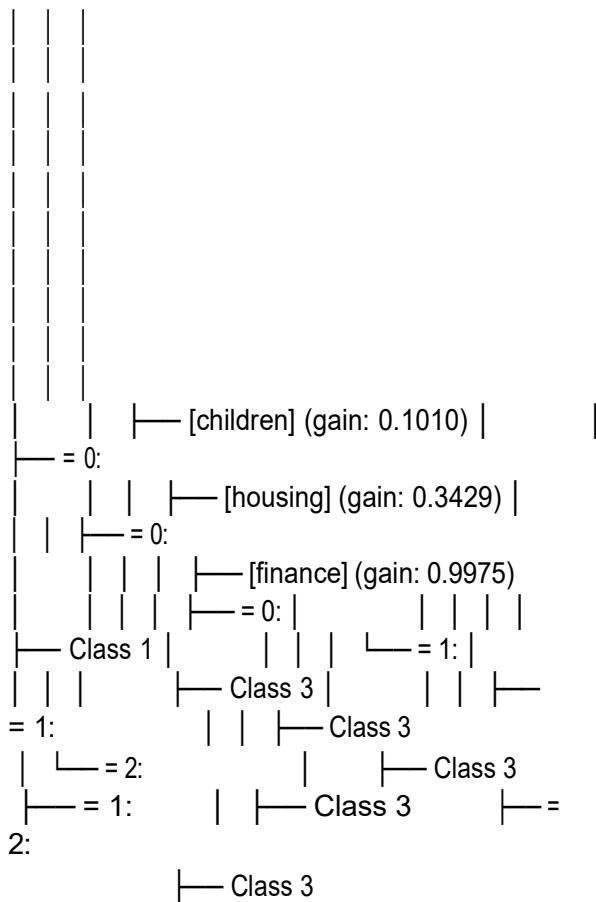
| | |

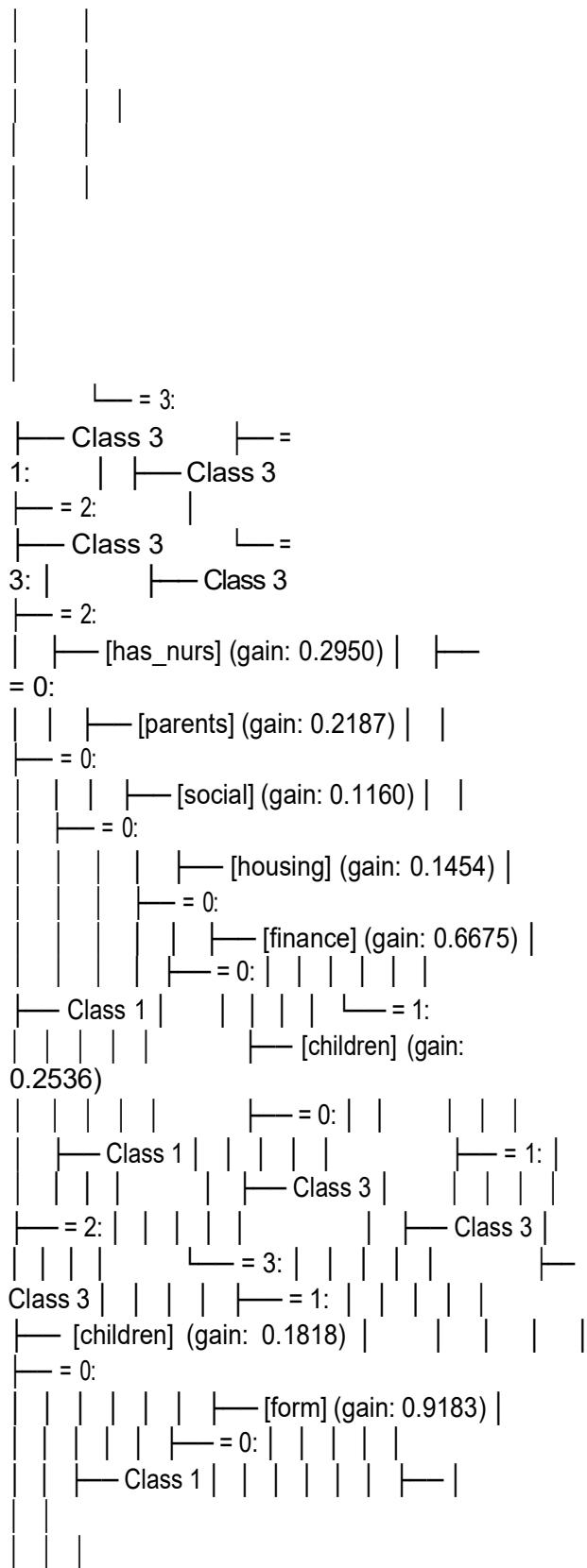
=  
└─ Class





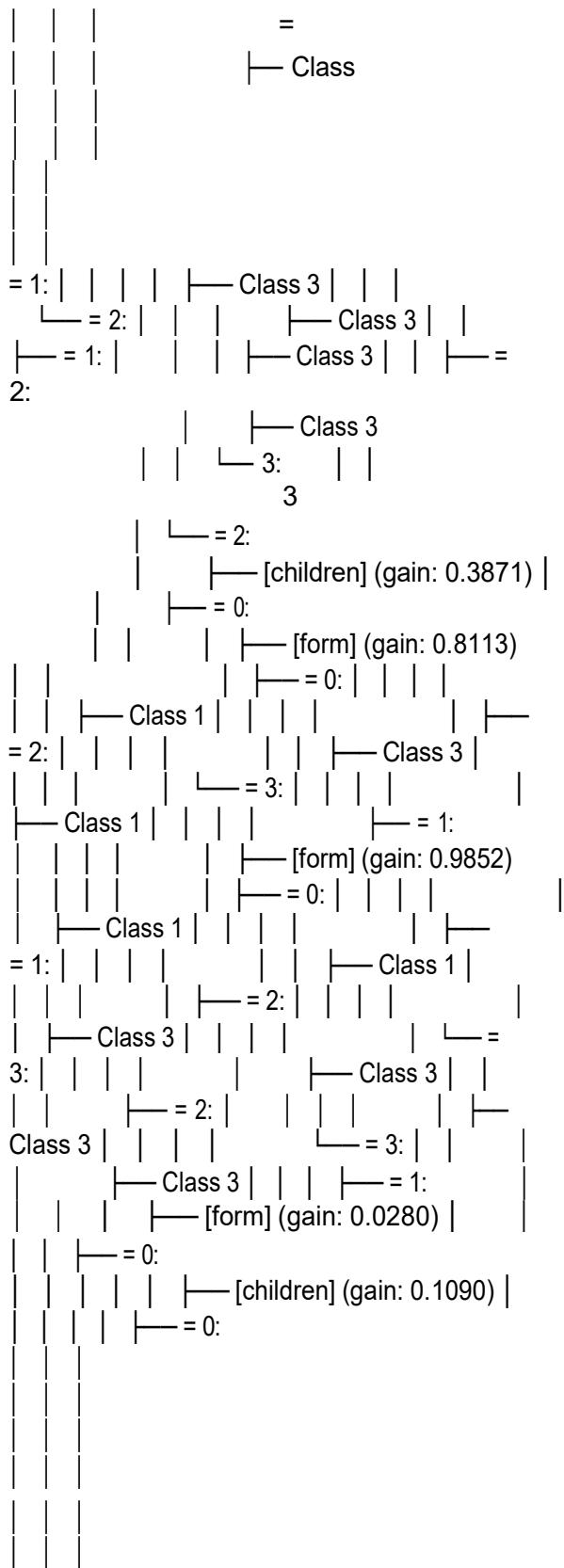
=  
└─ Class

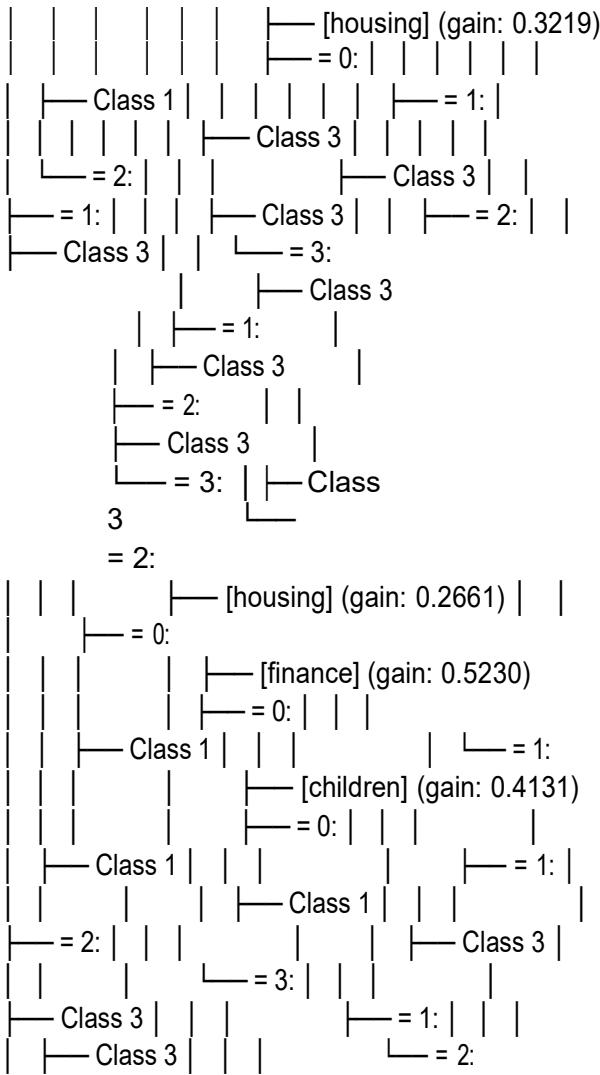
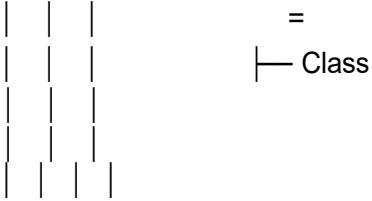


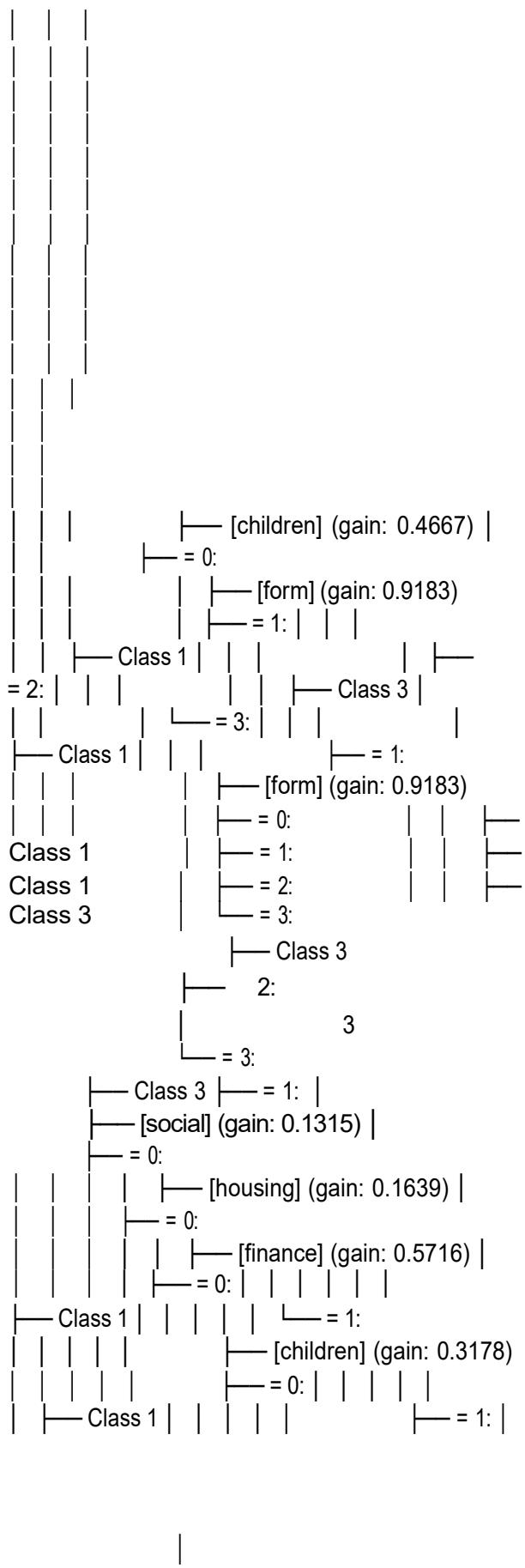


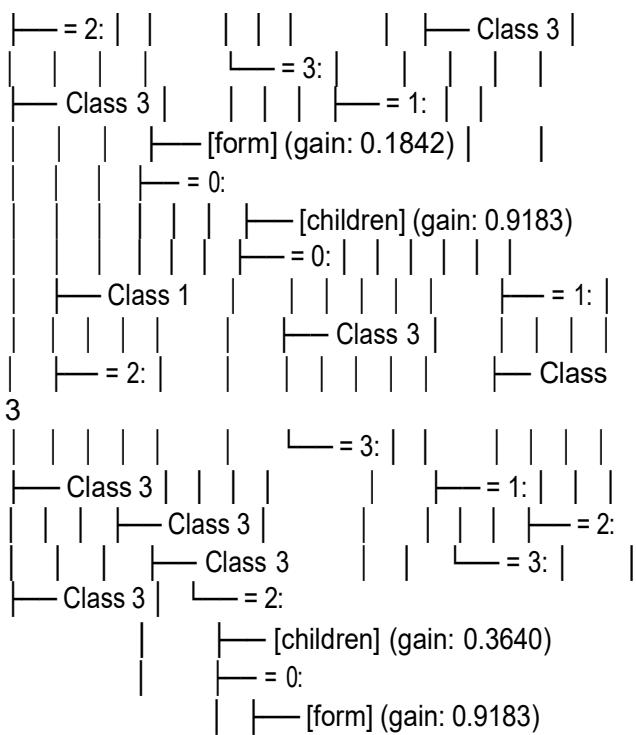
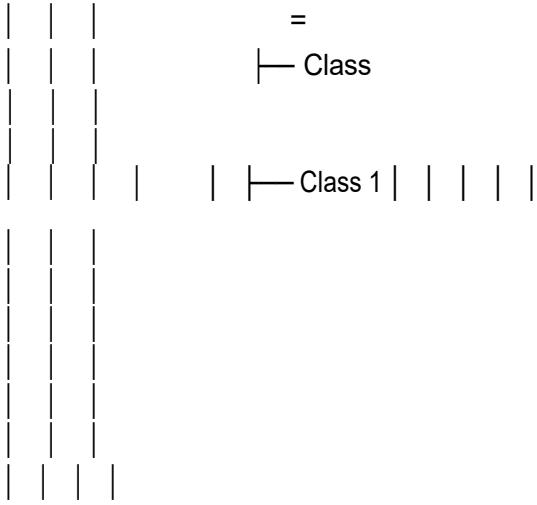
=  
└— Class



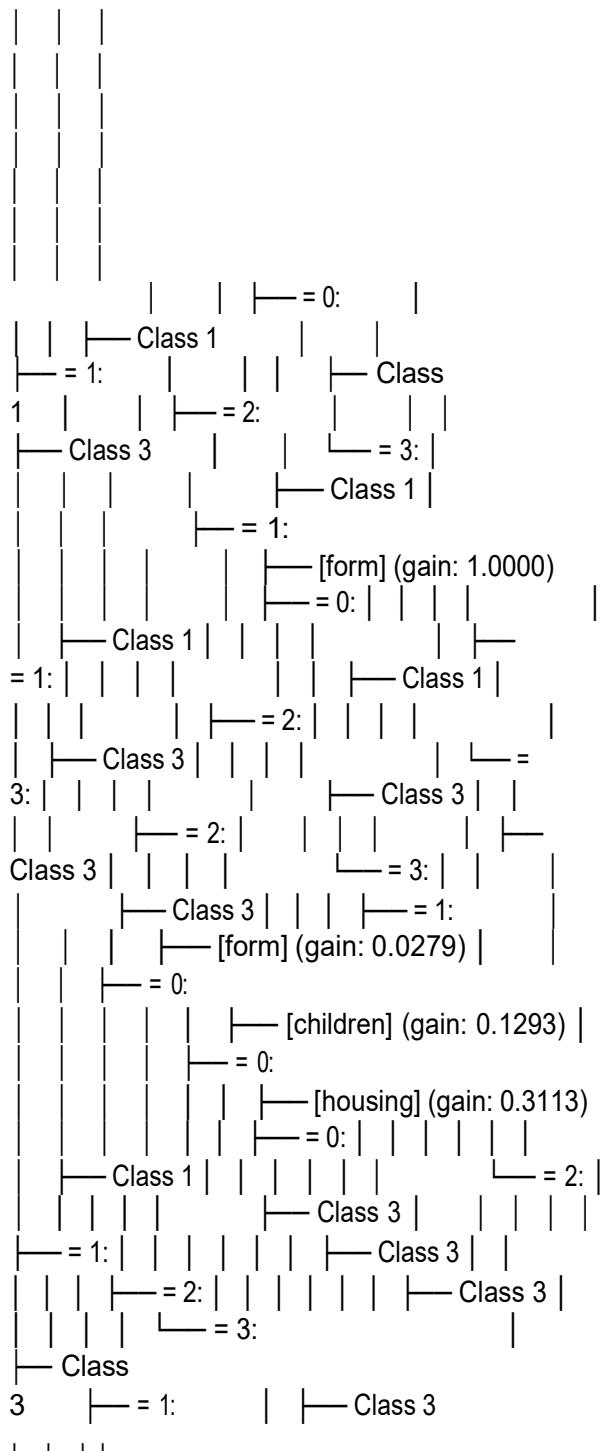


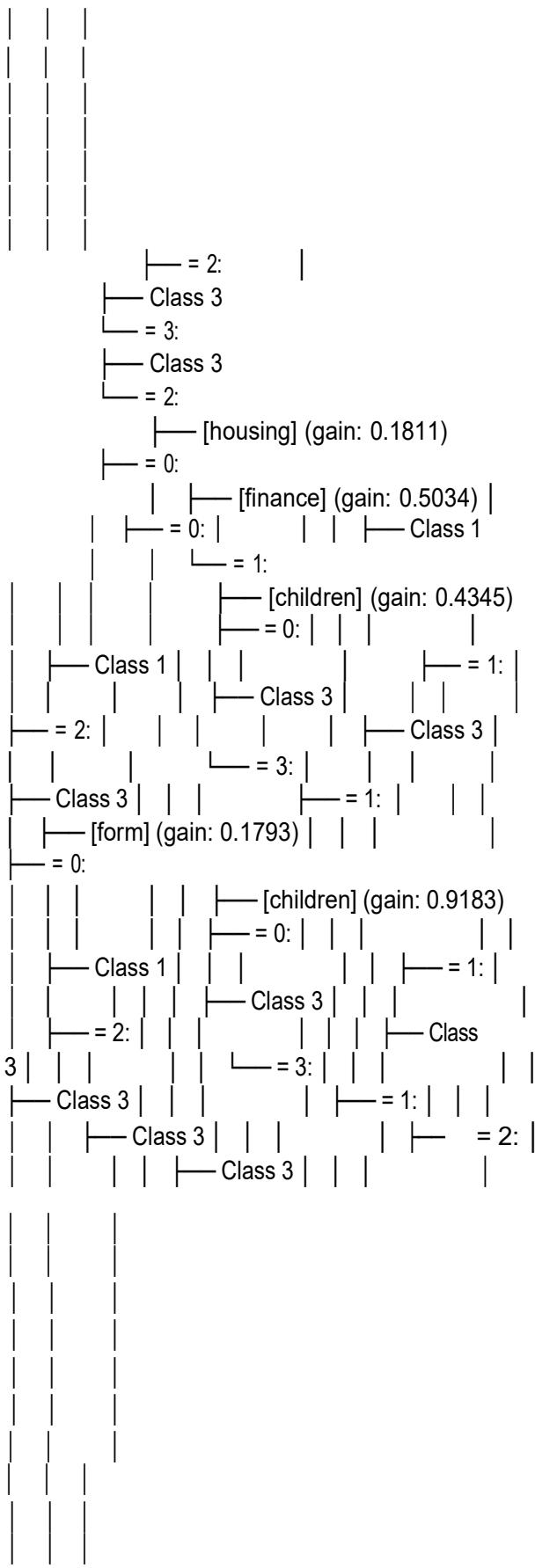


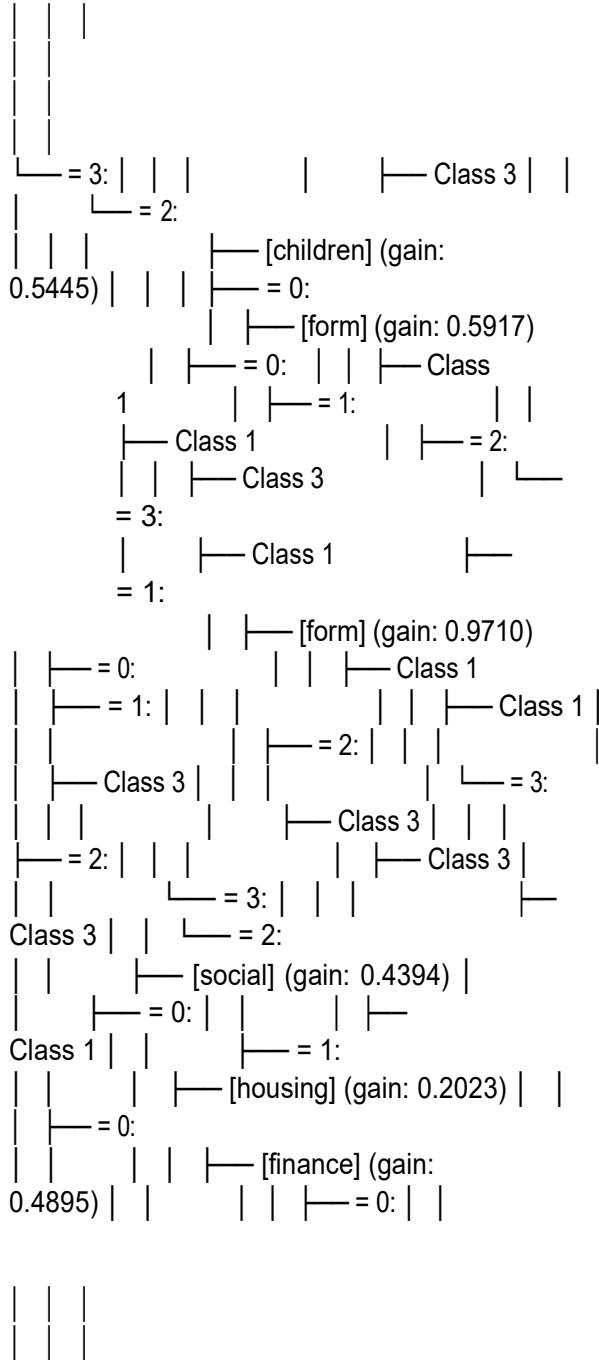


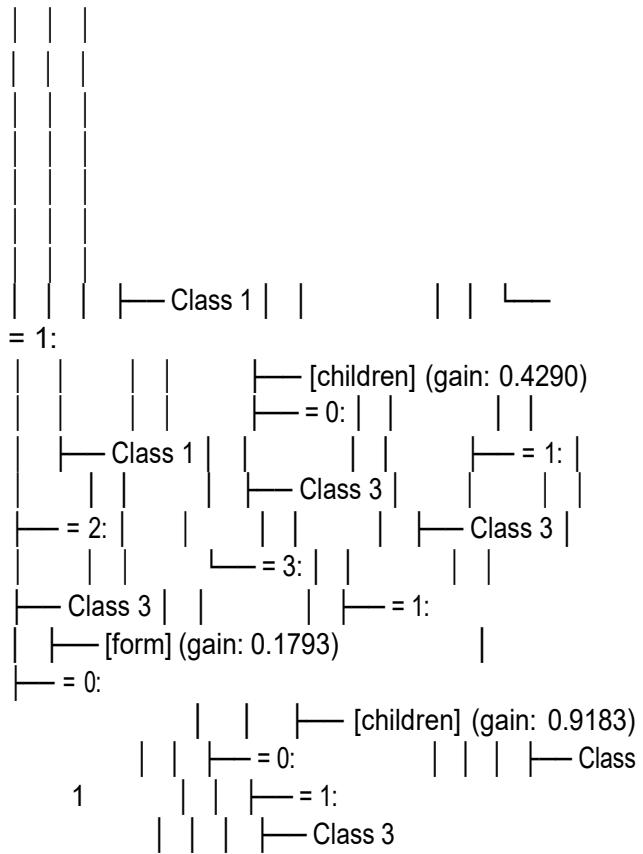




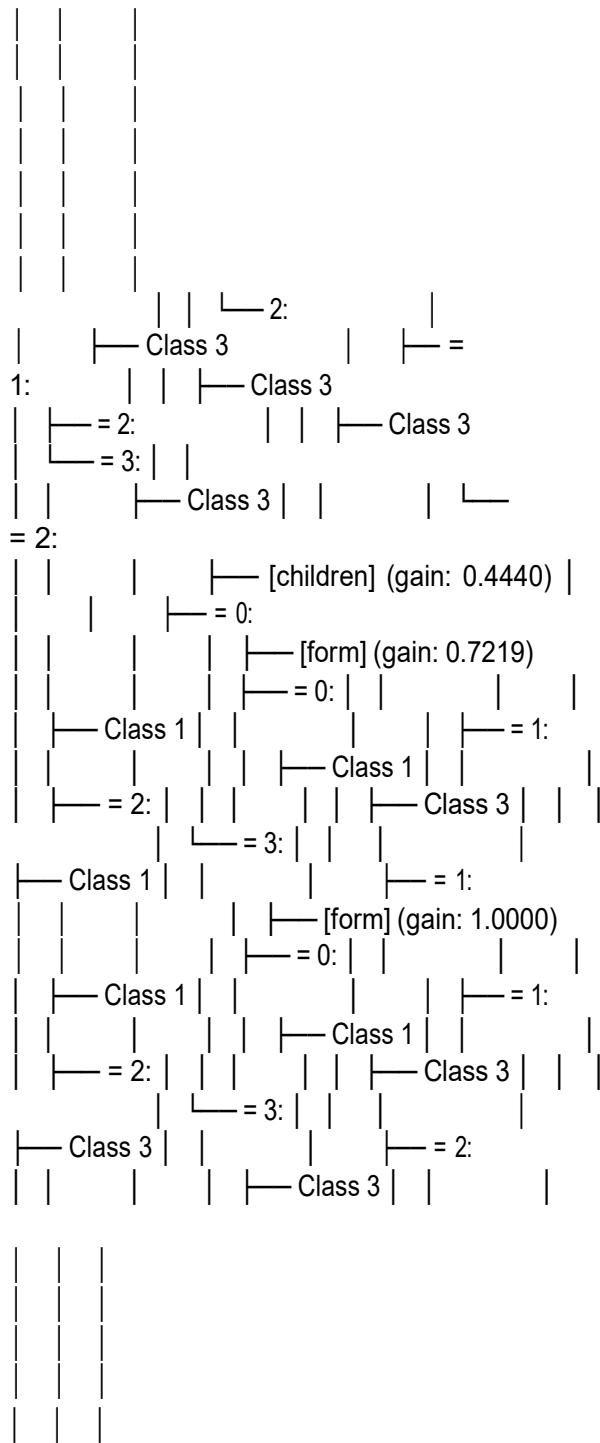


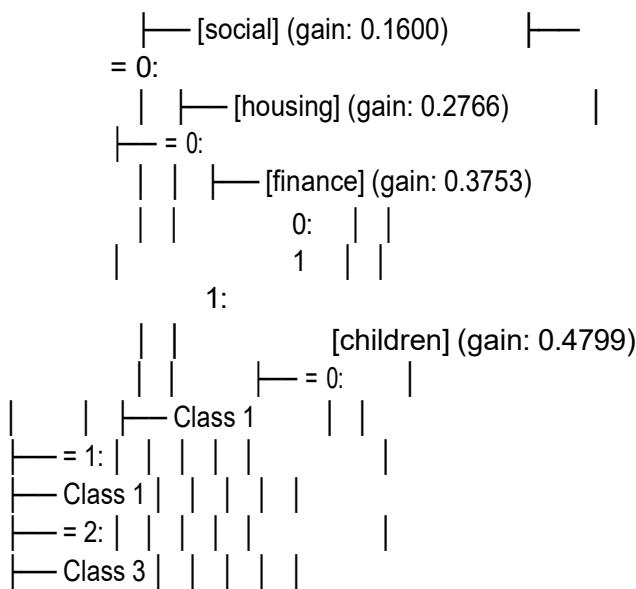
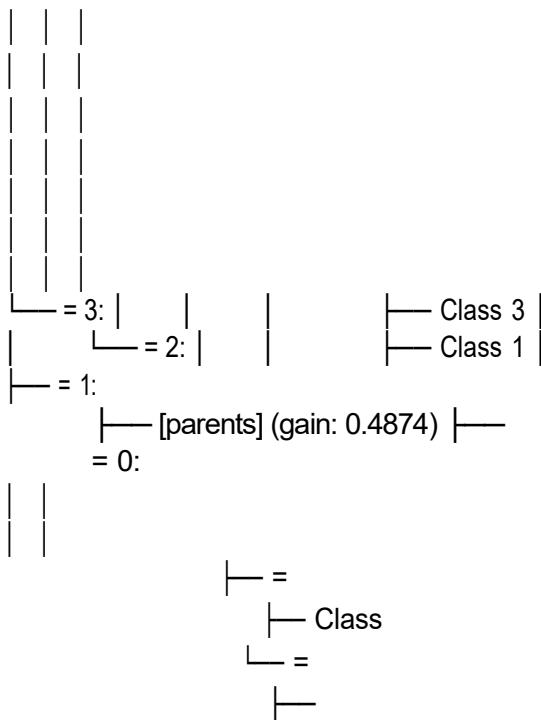




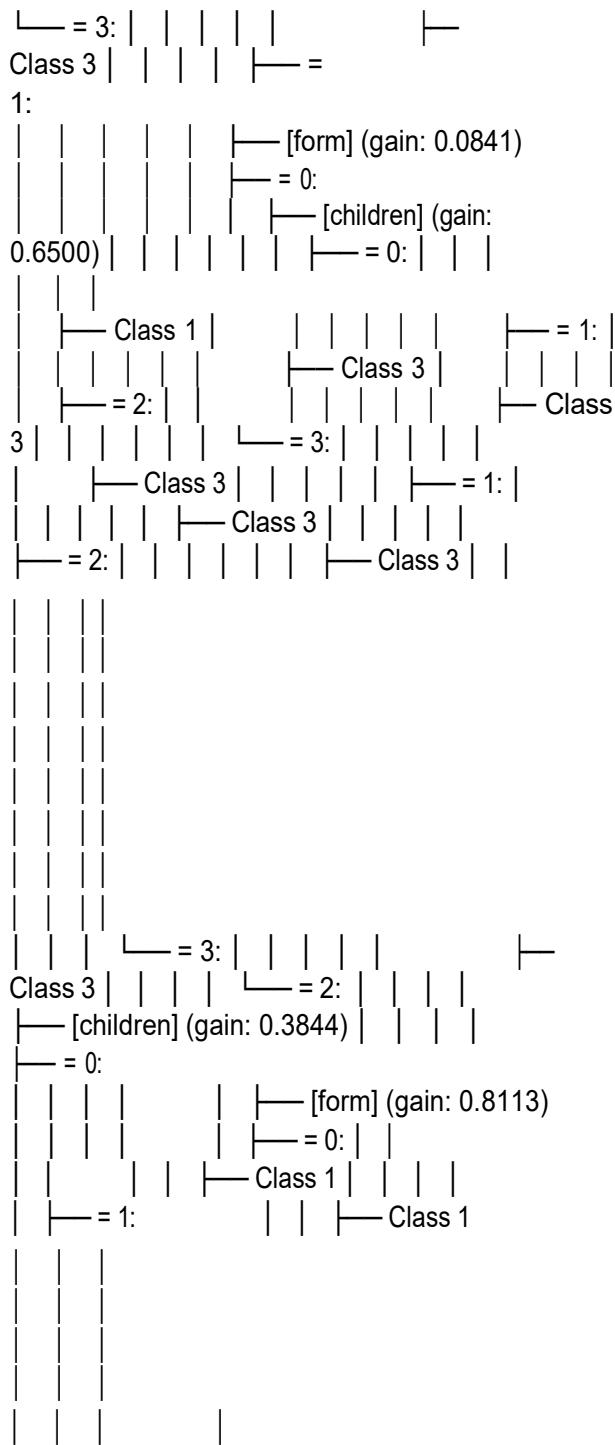


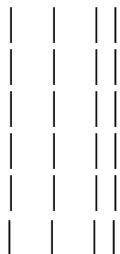
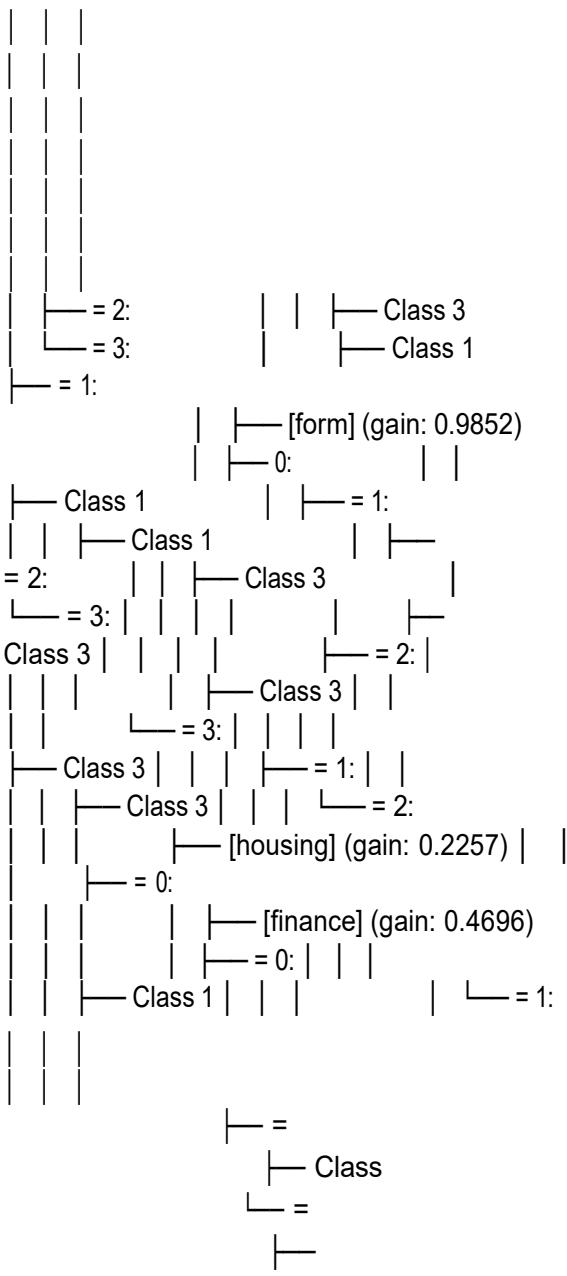
=



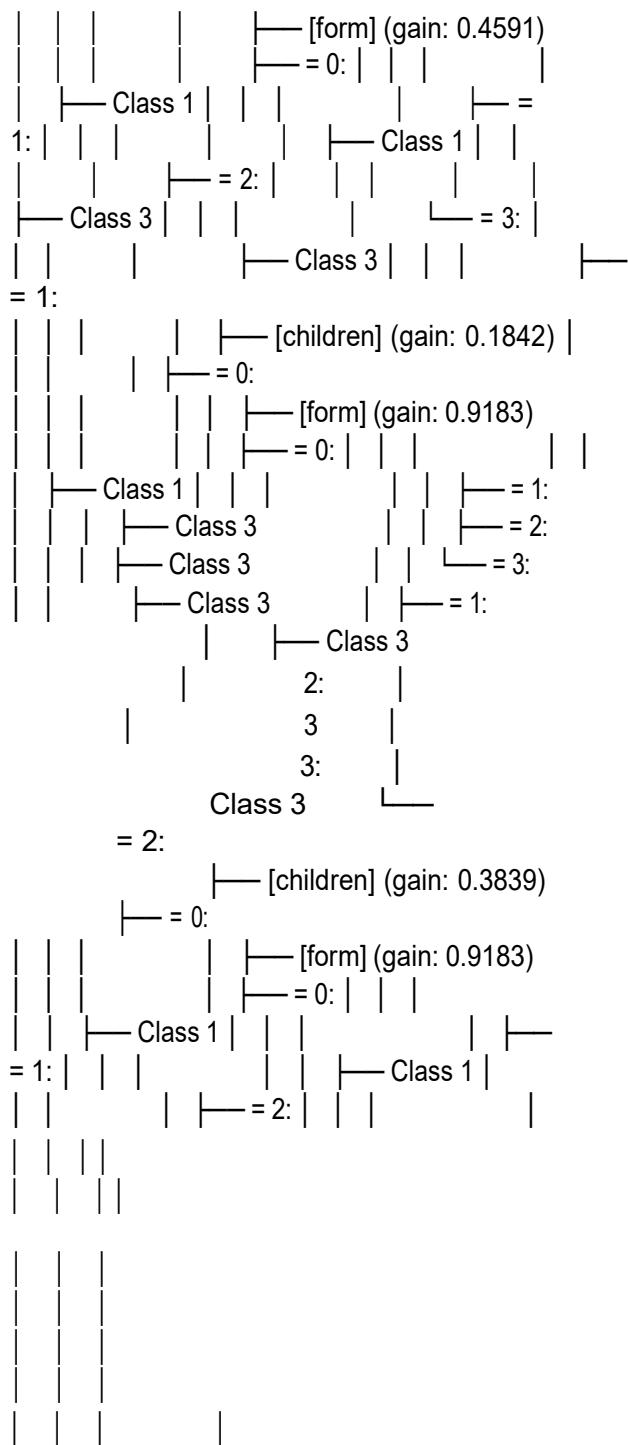


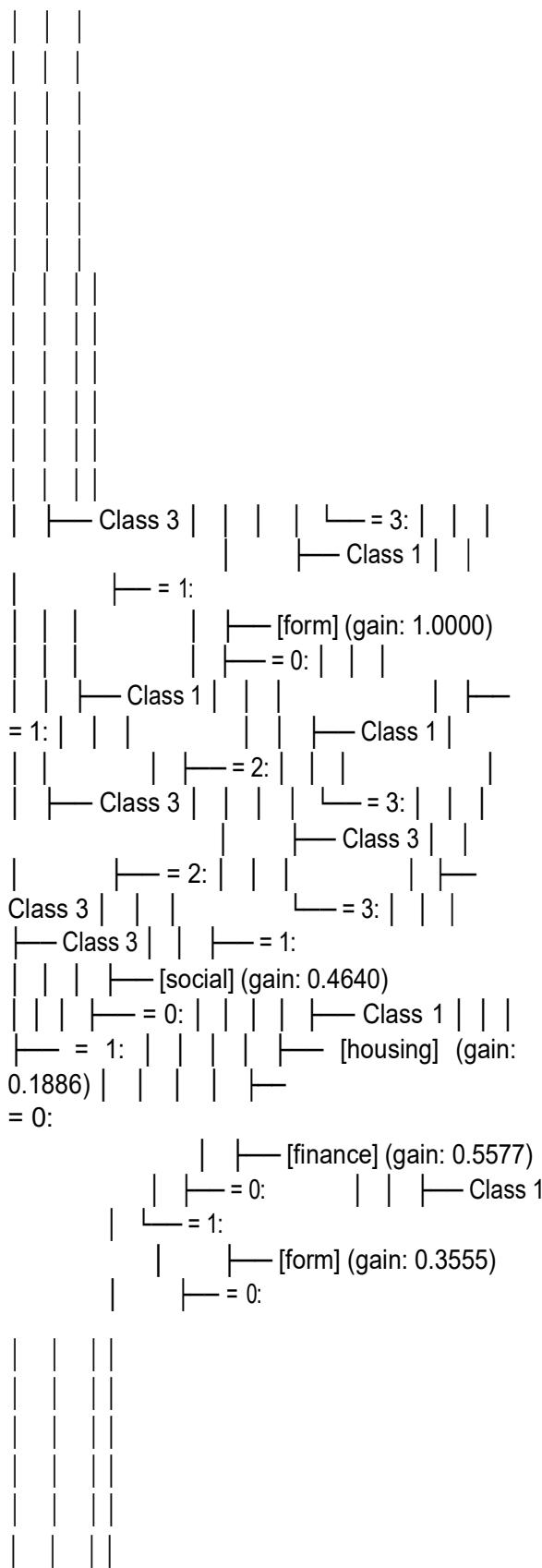
=





=



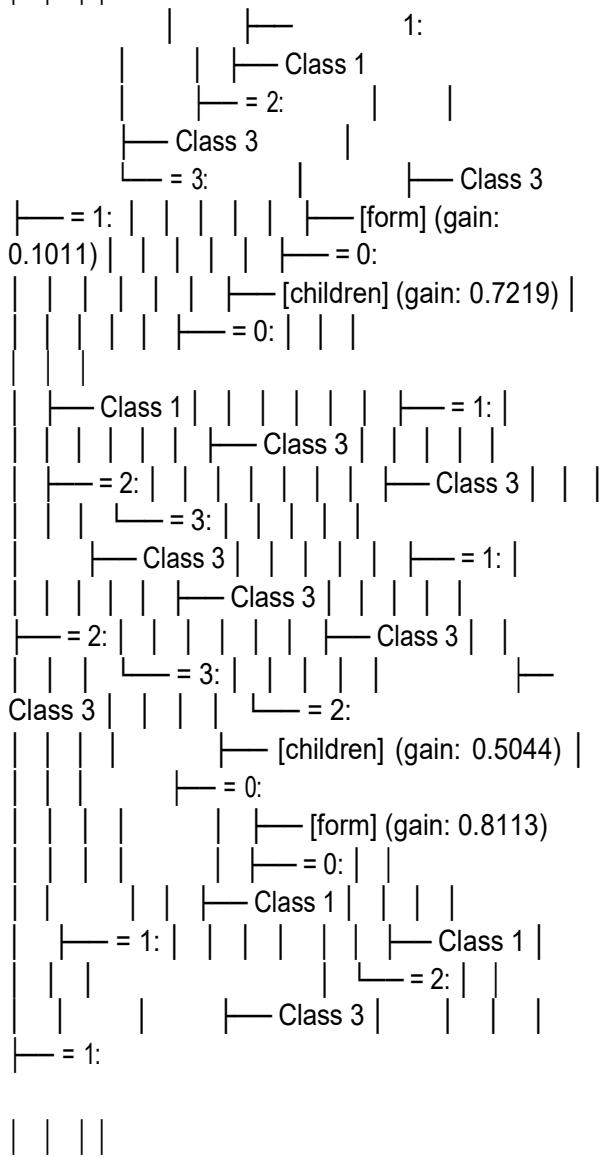
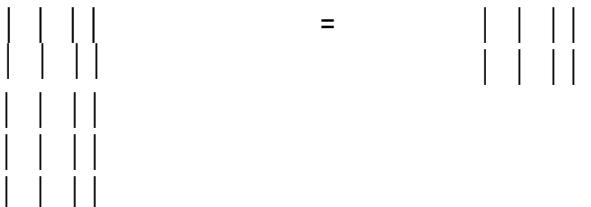


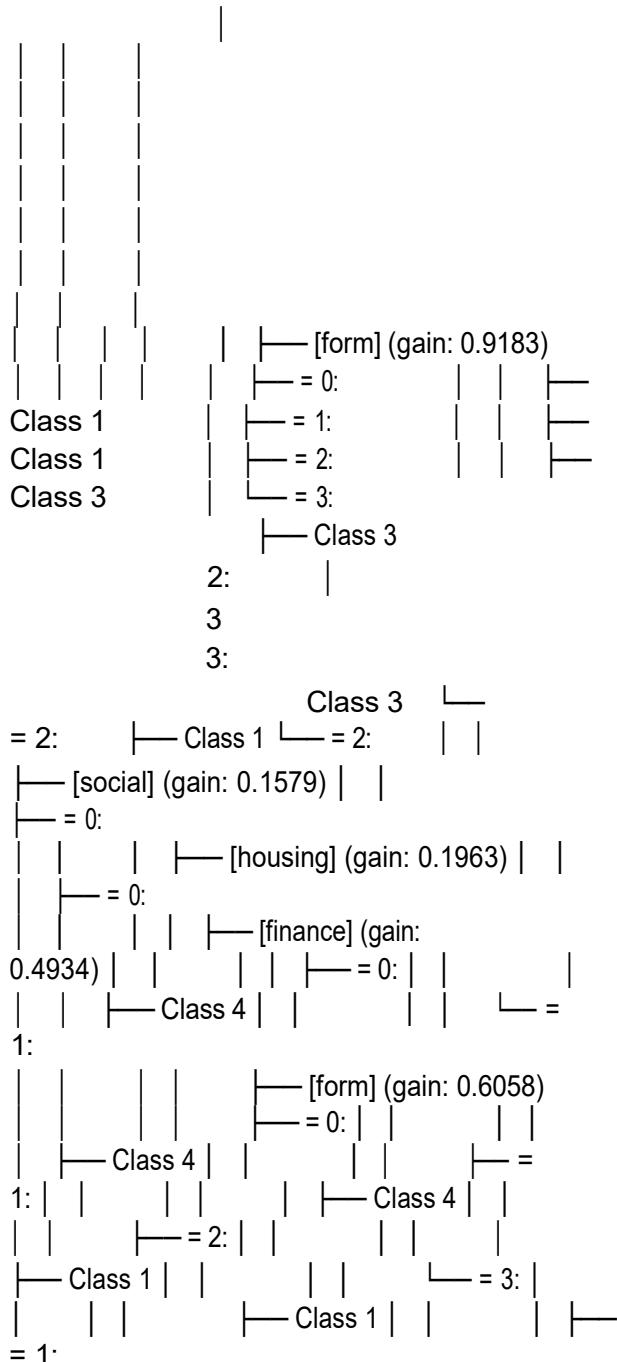
=

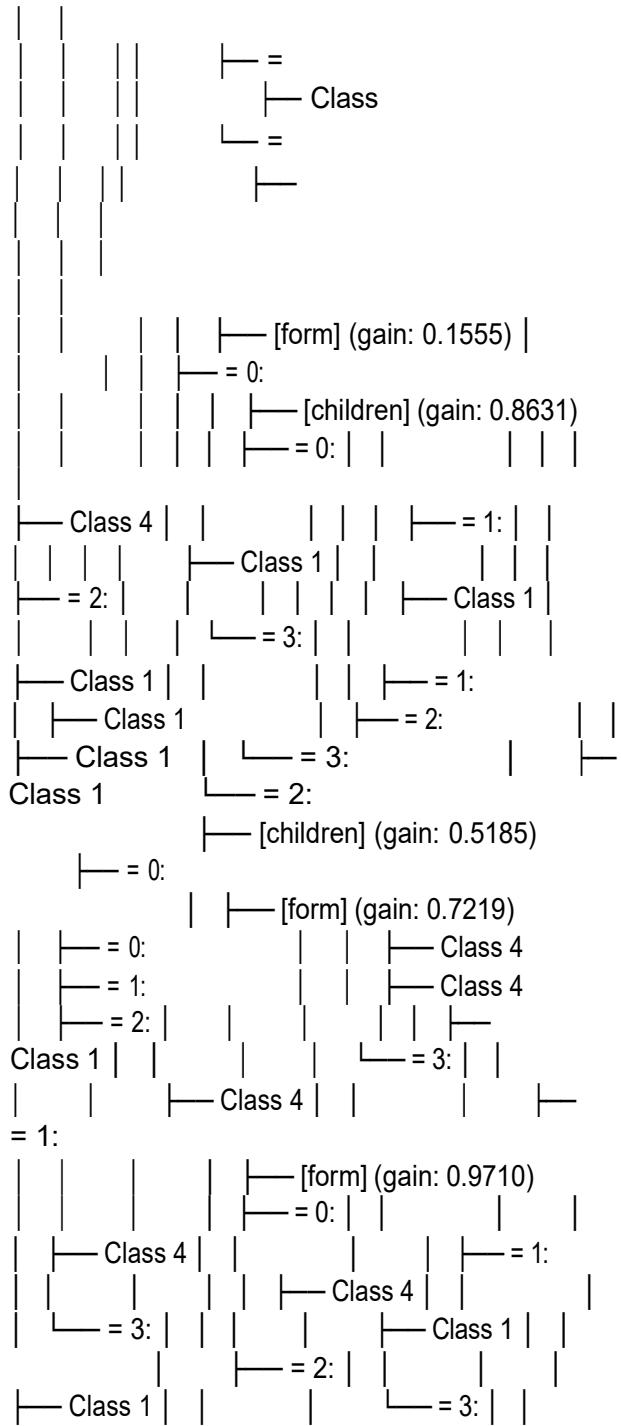
| | Class 3

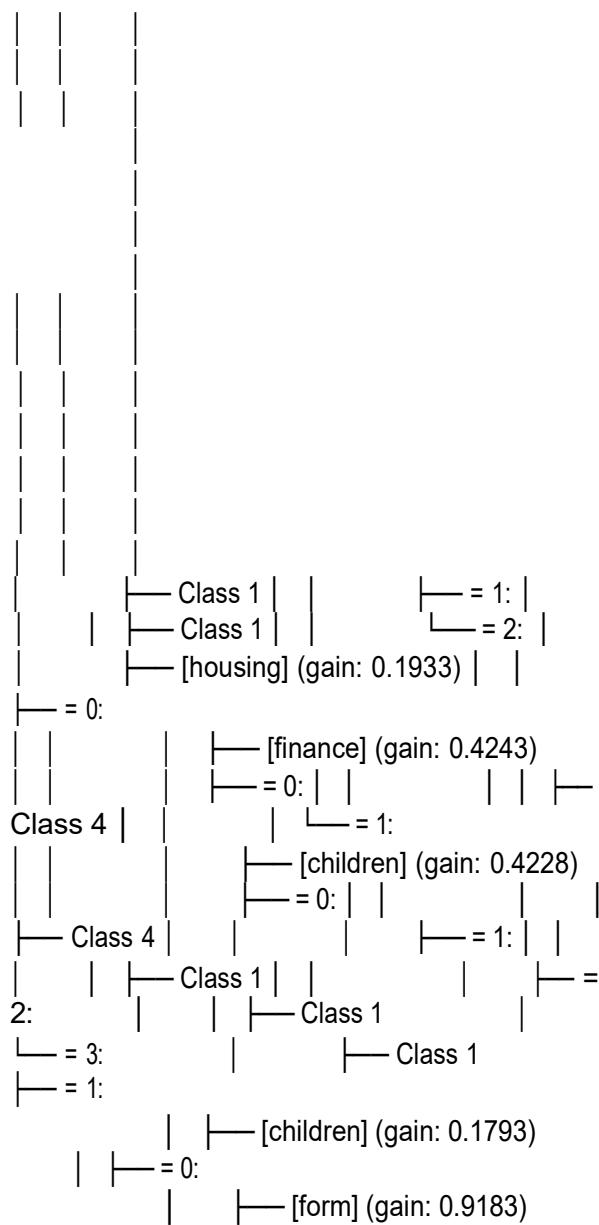
| | |

| | | |



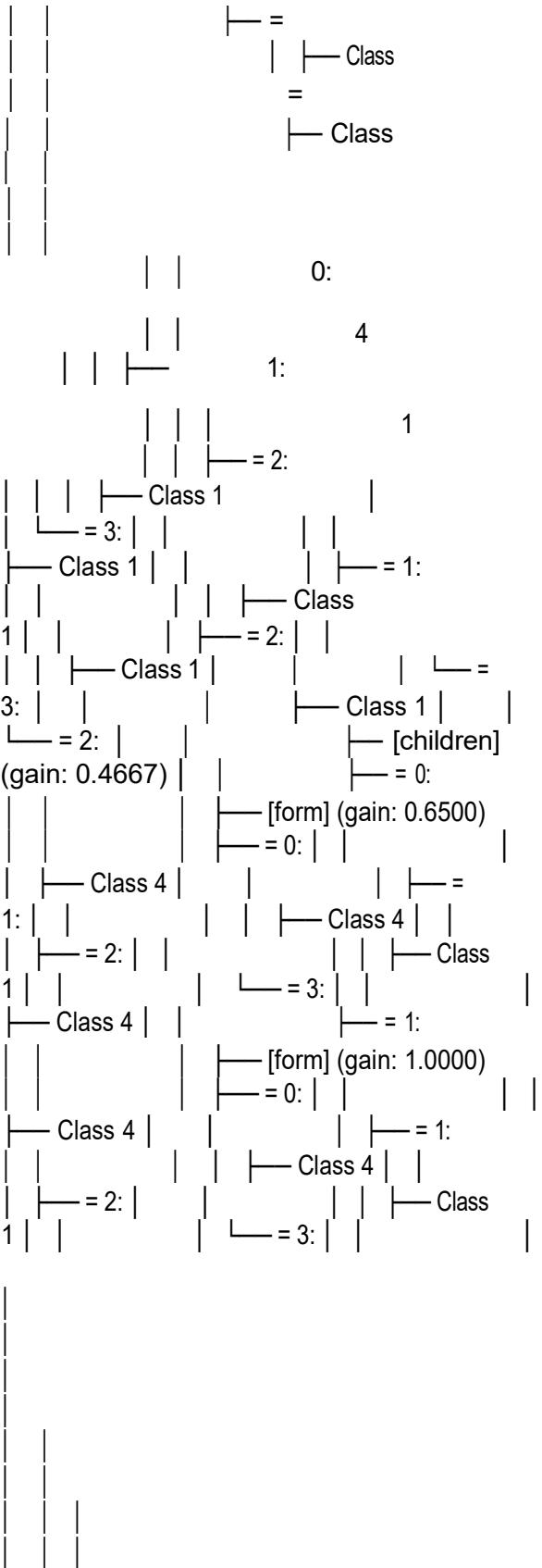


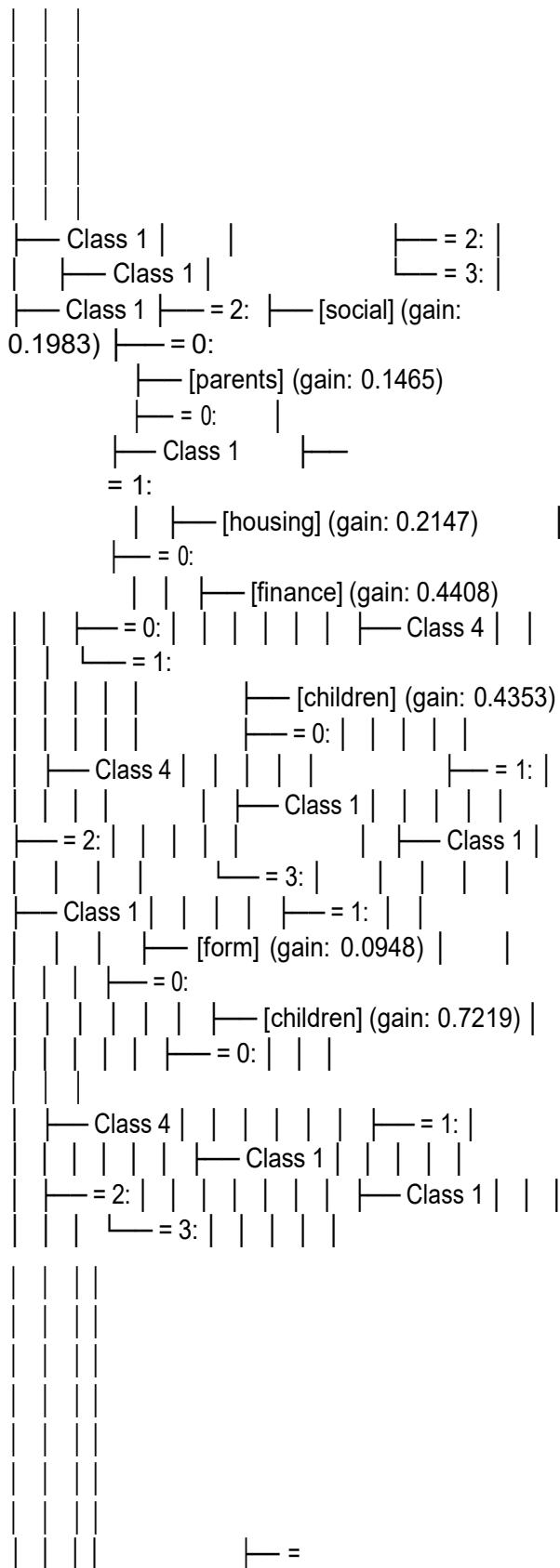


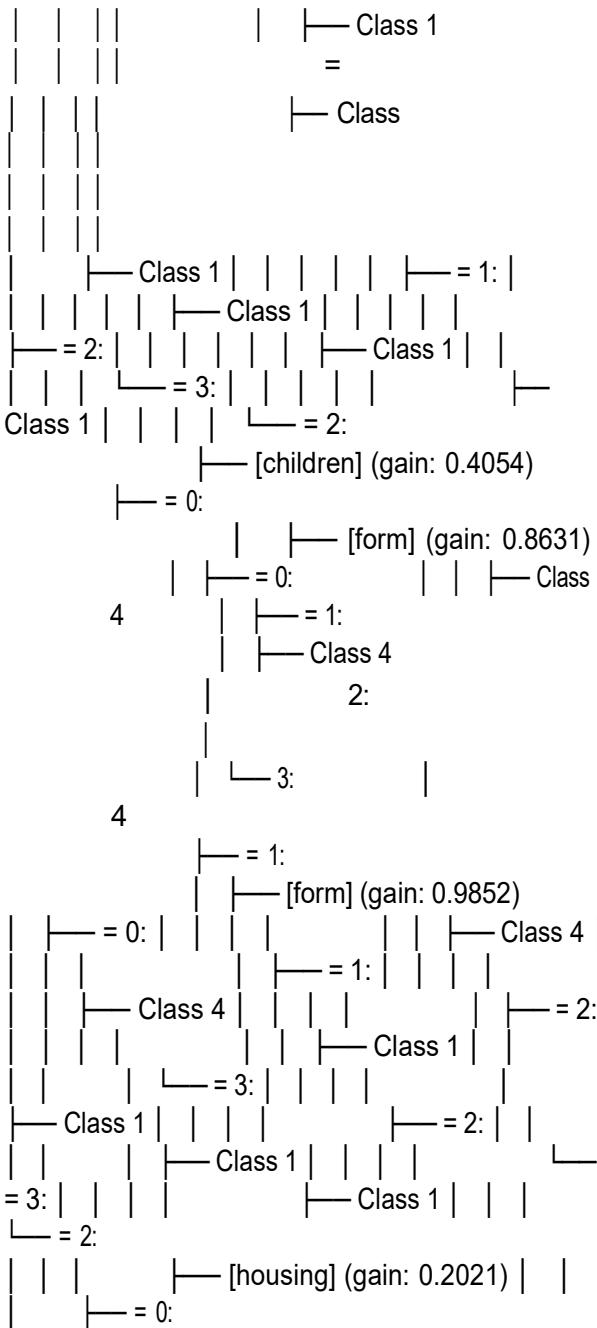


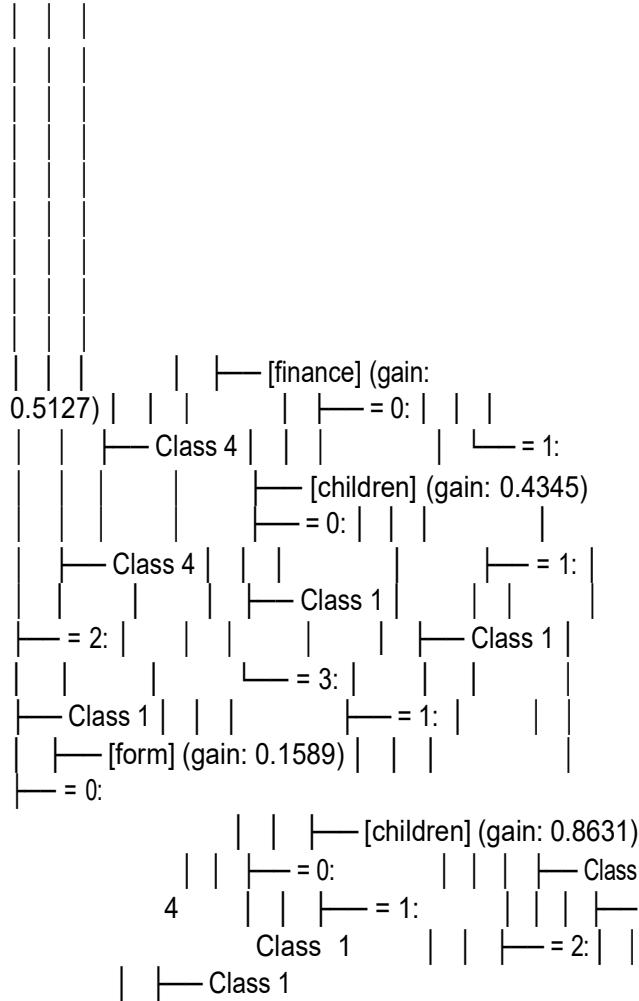
||

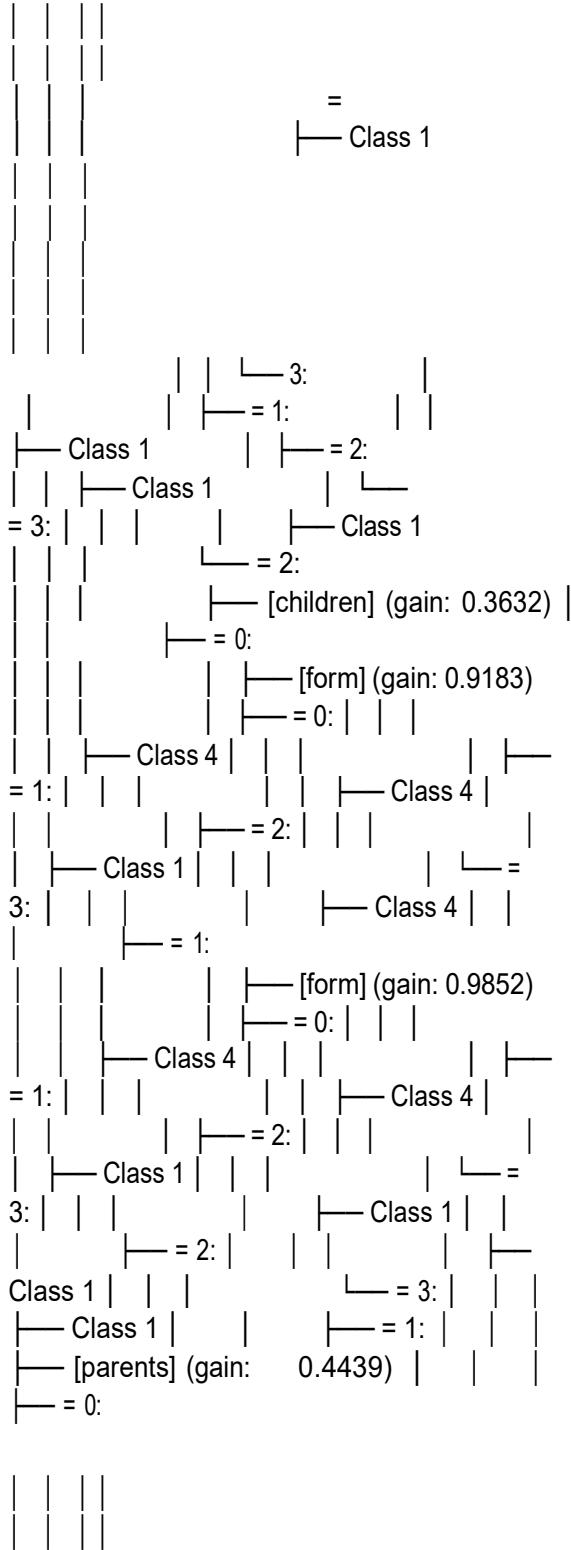
||



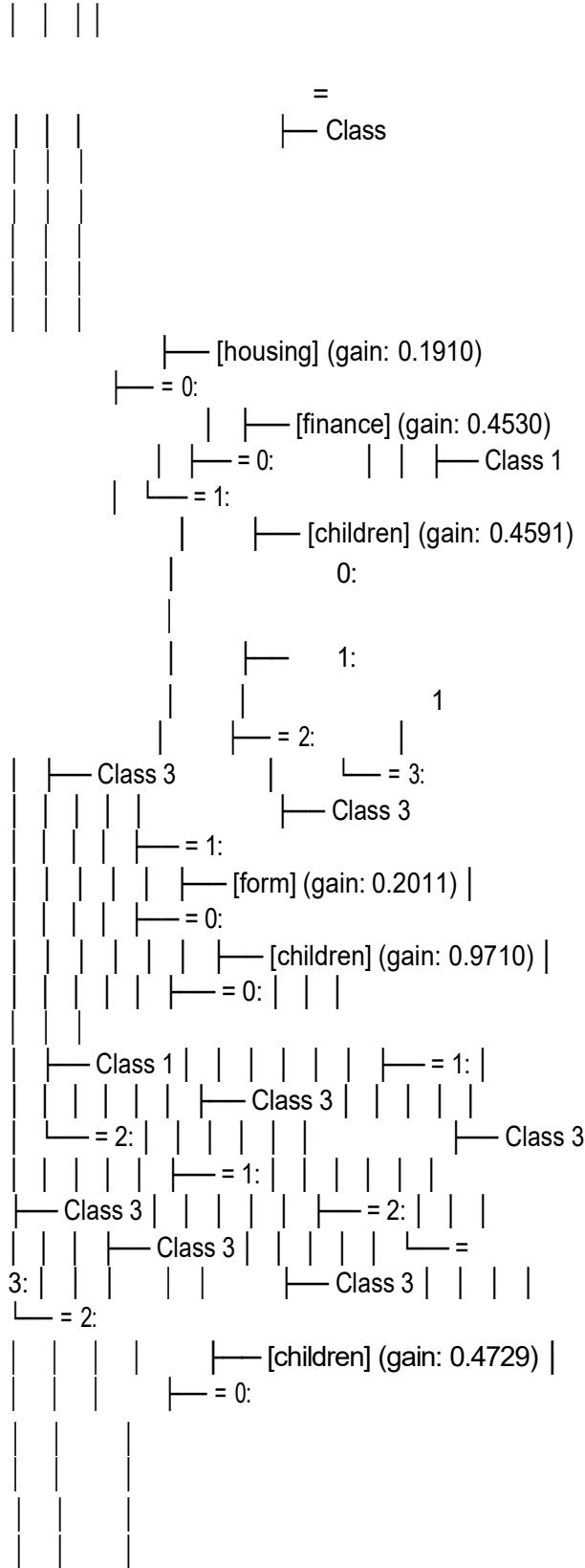




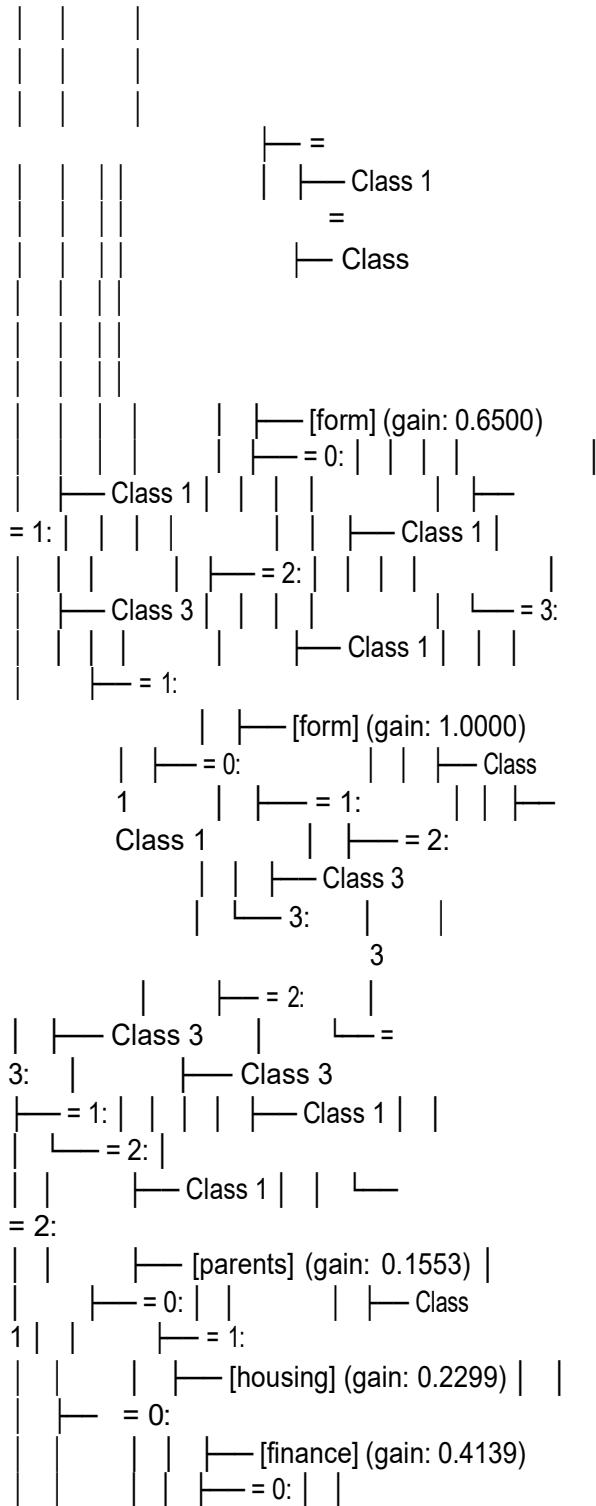


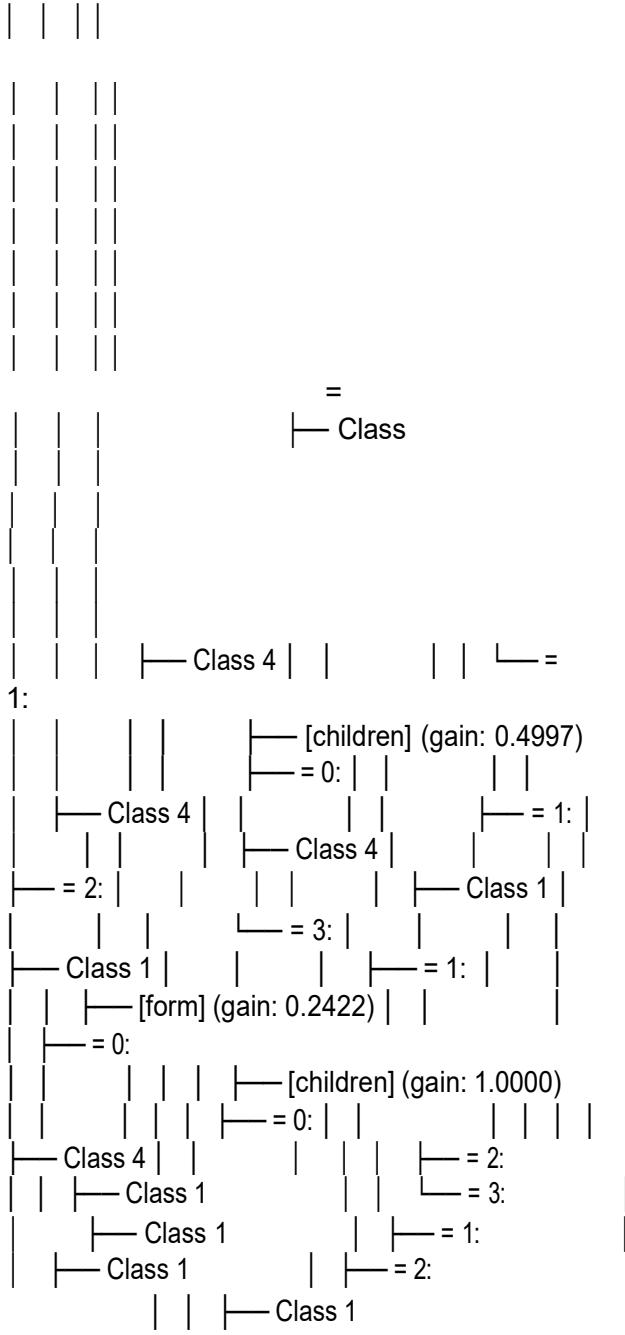






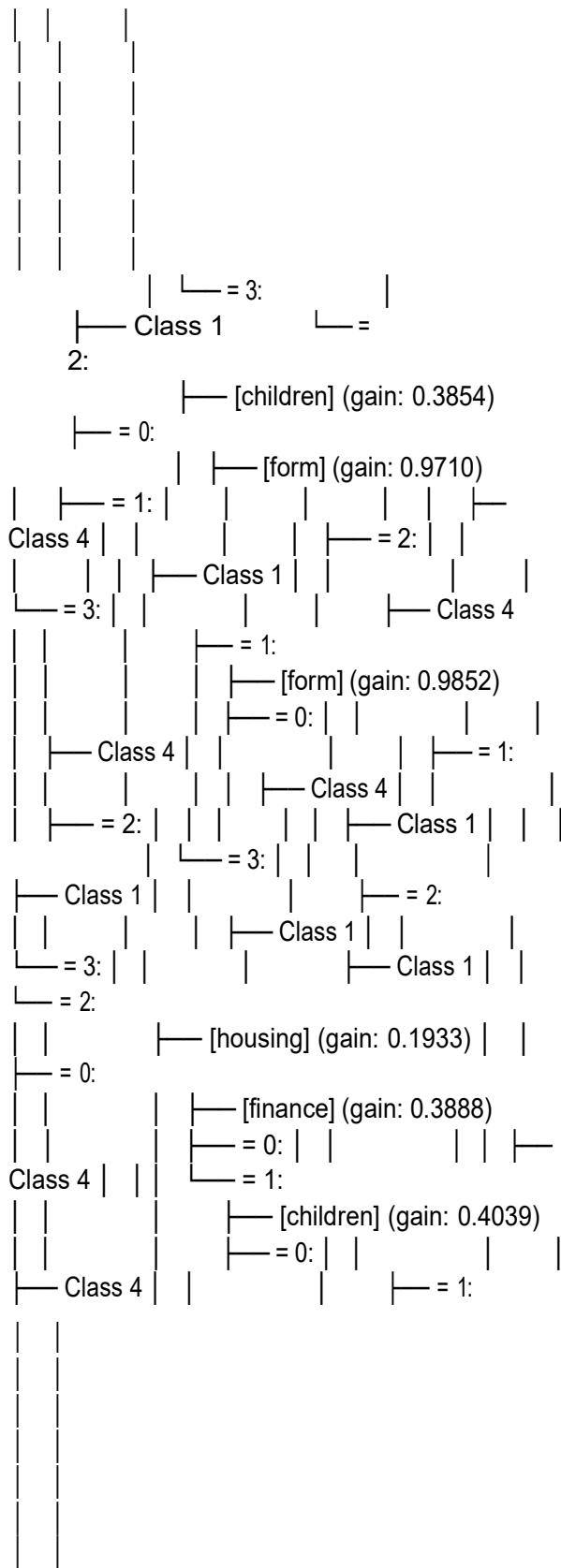
| | ||

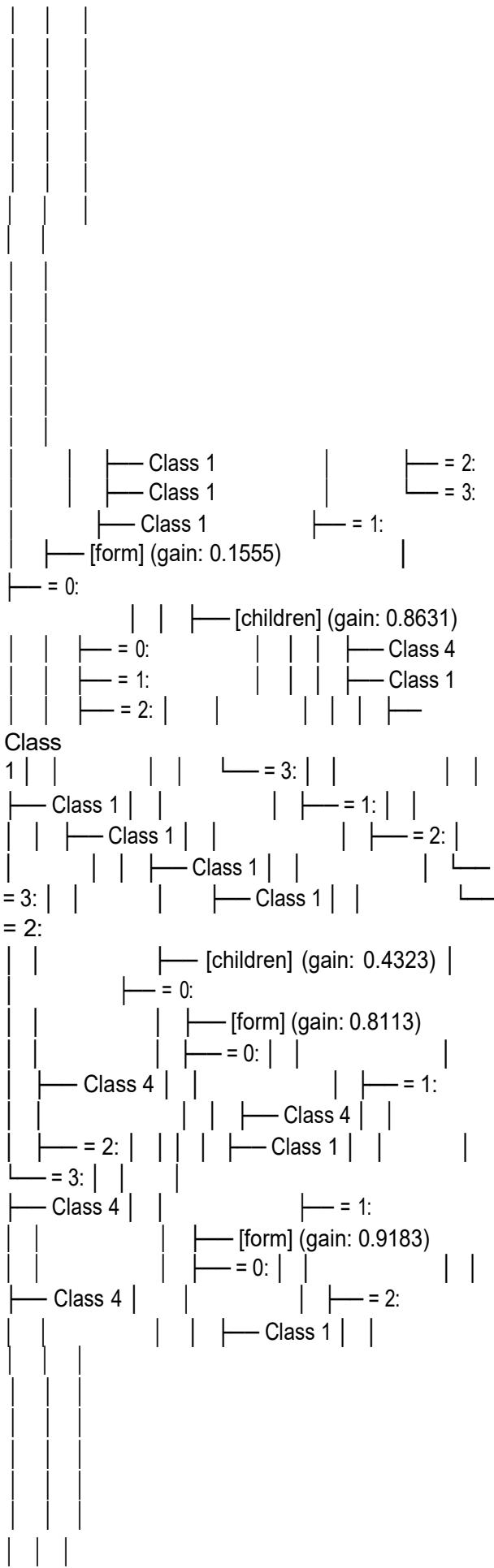


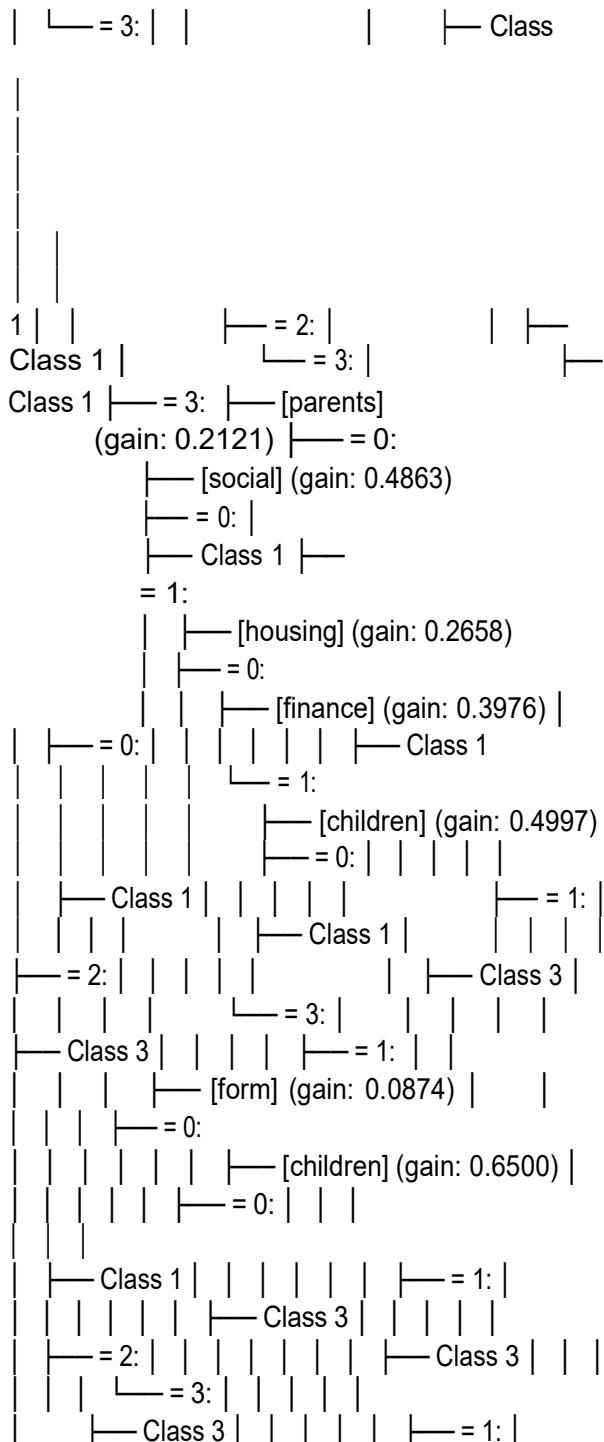


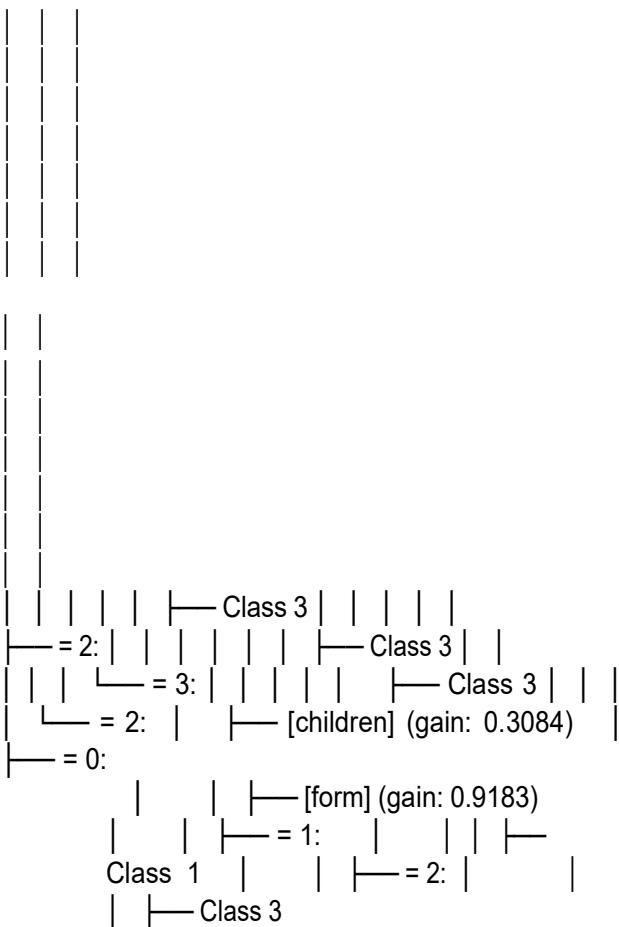
| | ||

|||

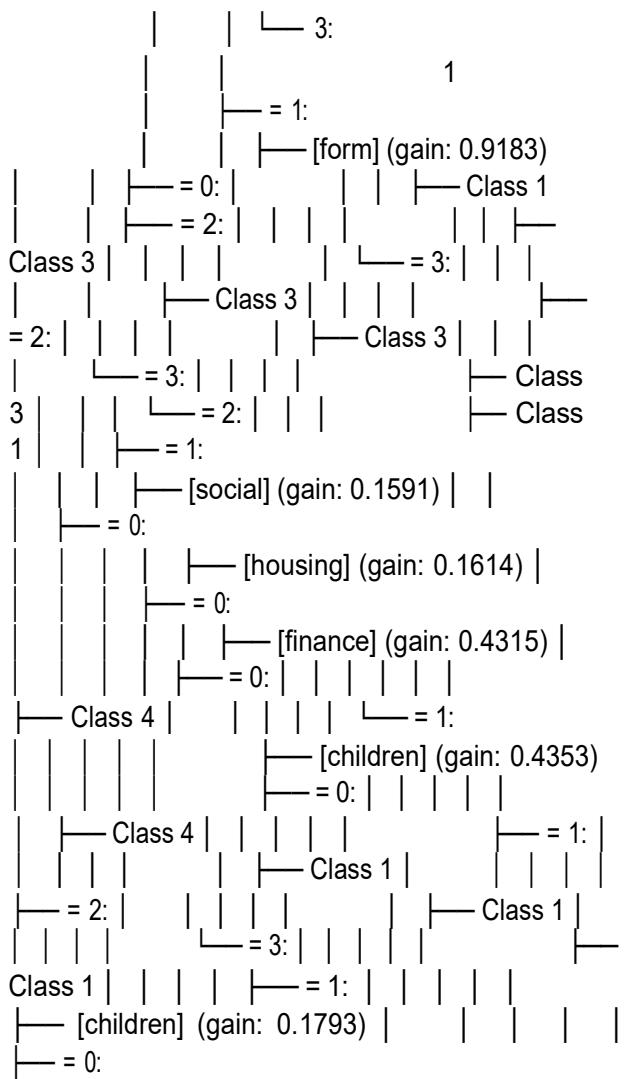








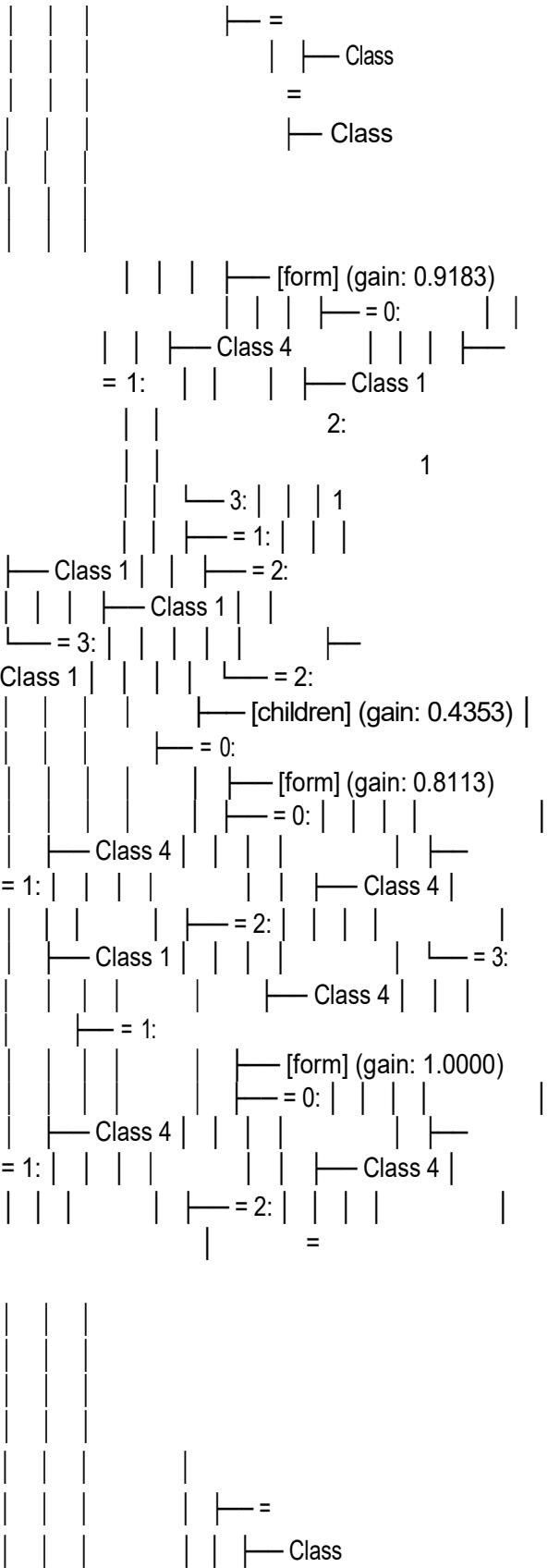
=  
└ Class

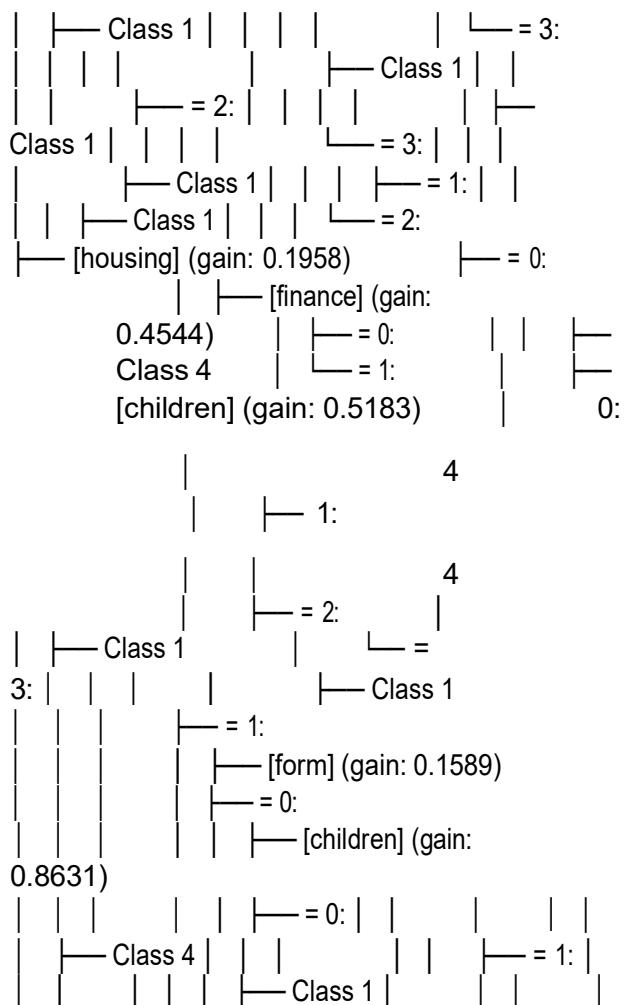
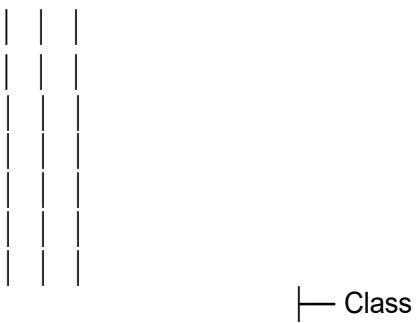


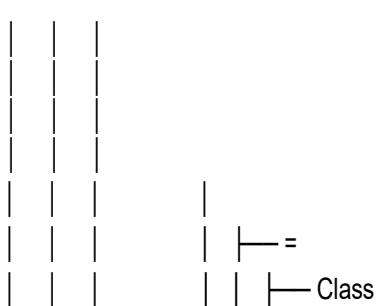
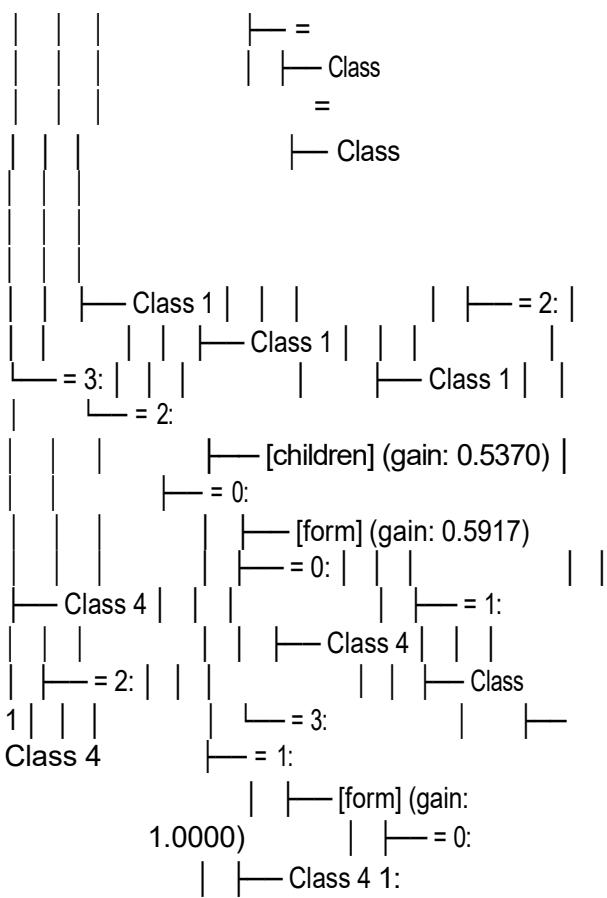
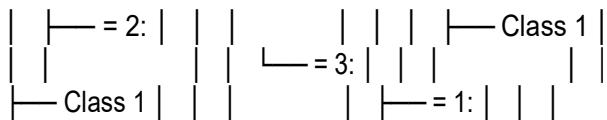


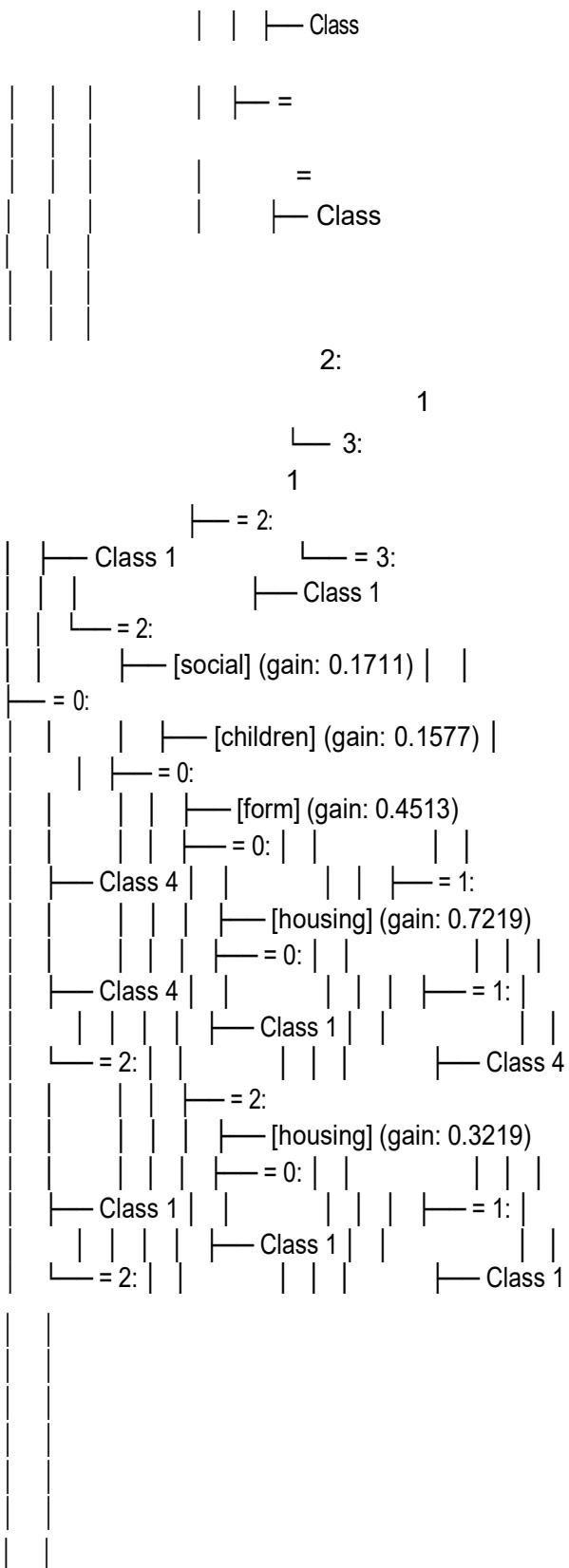
|  
|    =  
|    |— Class

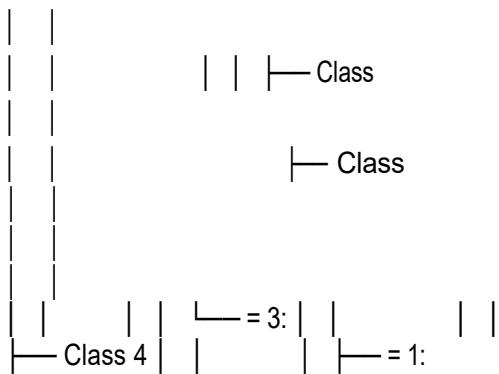




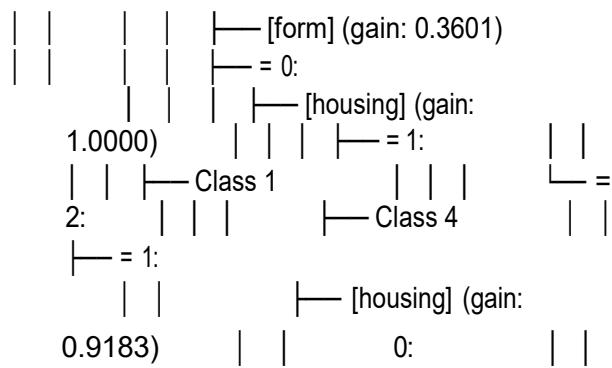




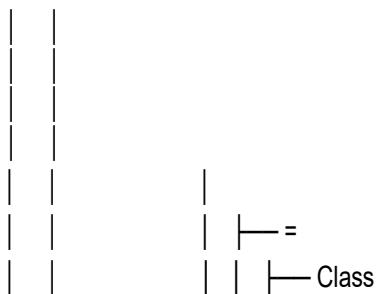


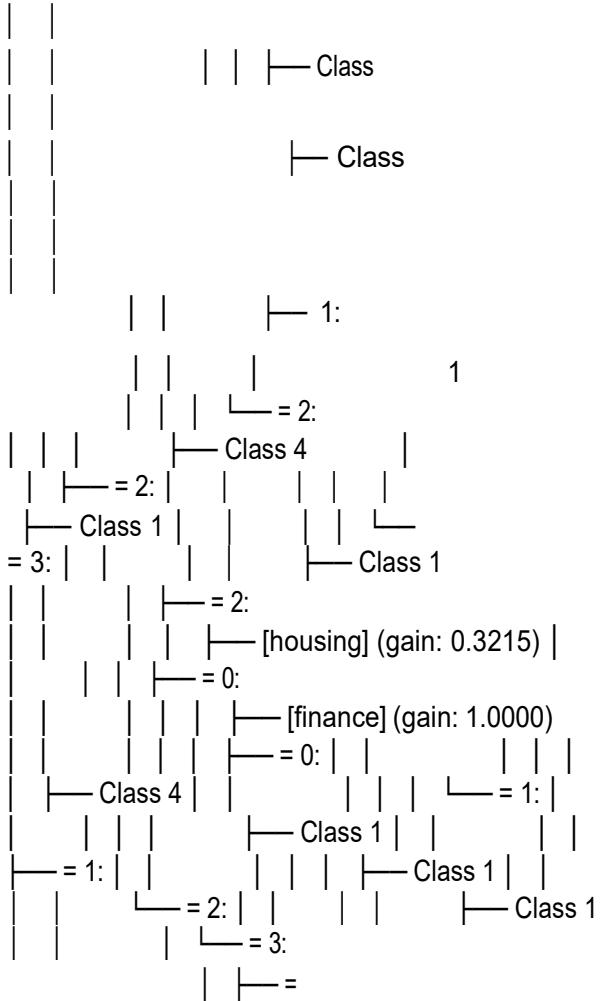


$|$   
 $|$   
 $|$   
 $|$   
 $|$   
 $=$   
 $=$   
 $=$

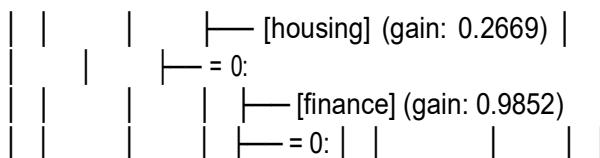


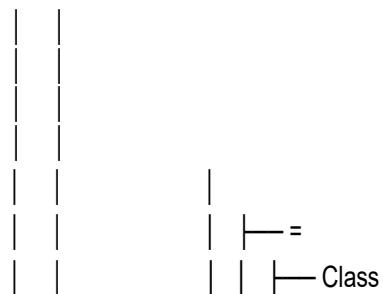
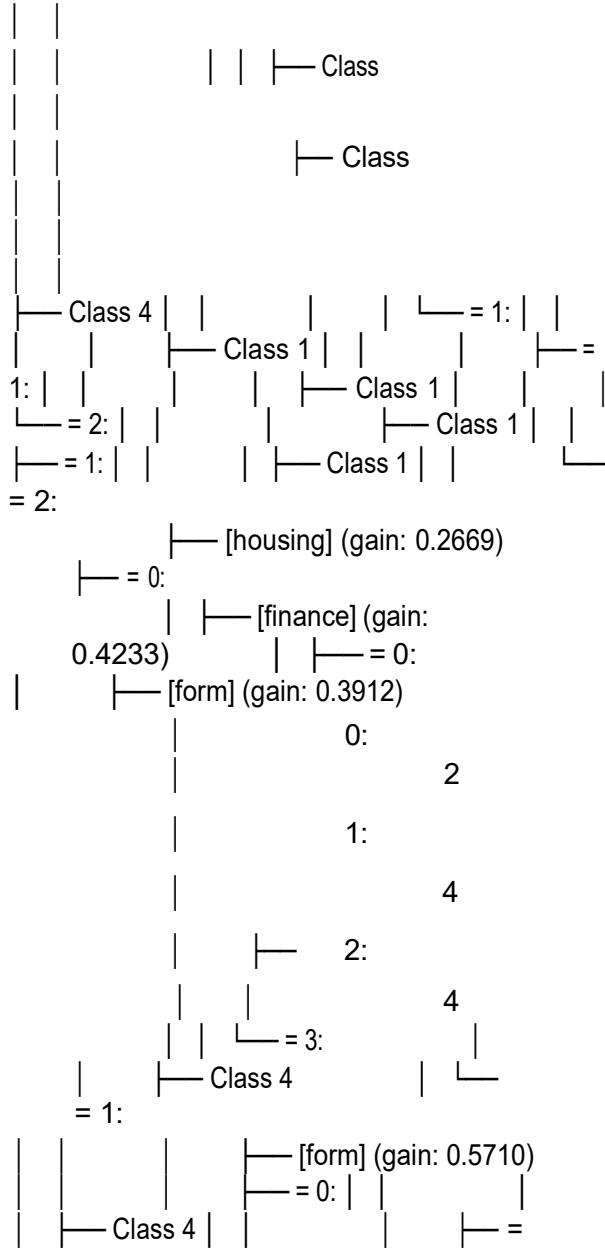
4

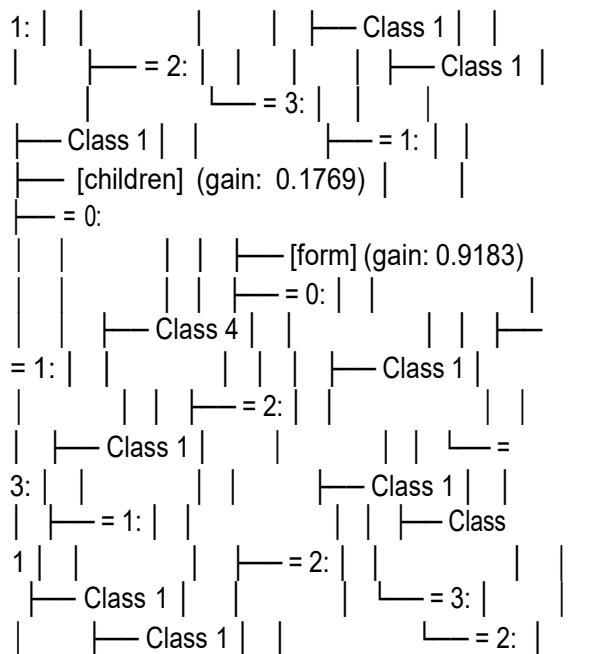
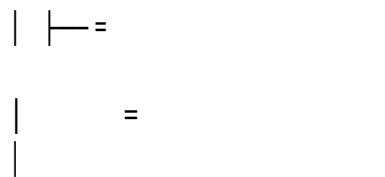
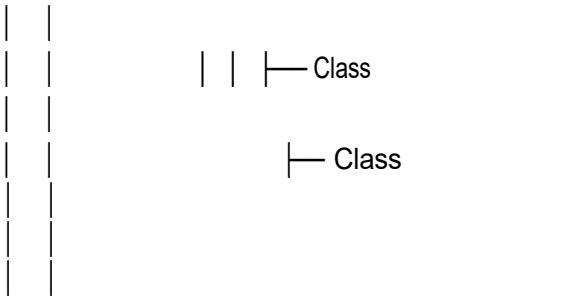


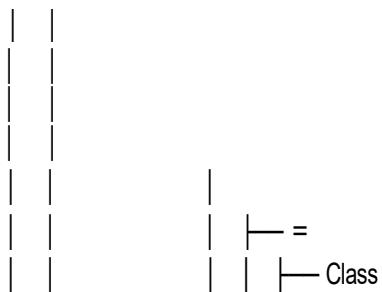
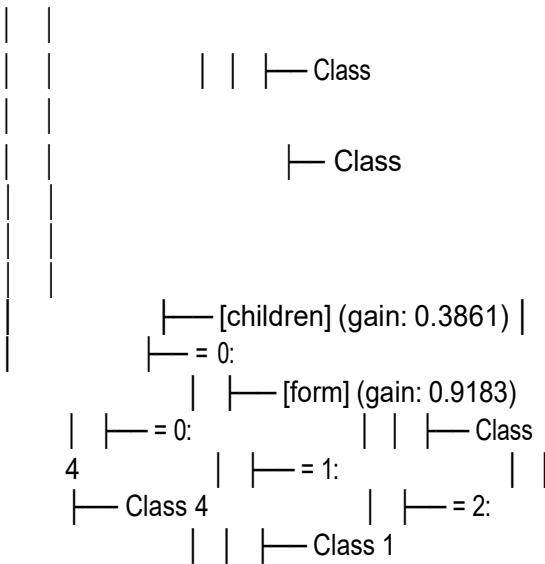


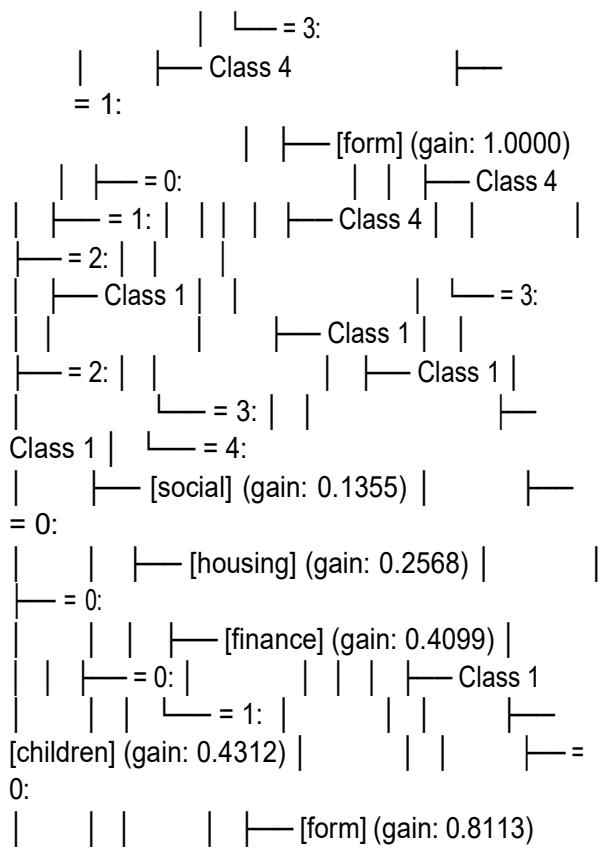
=

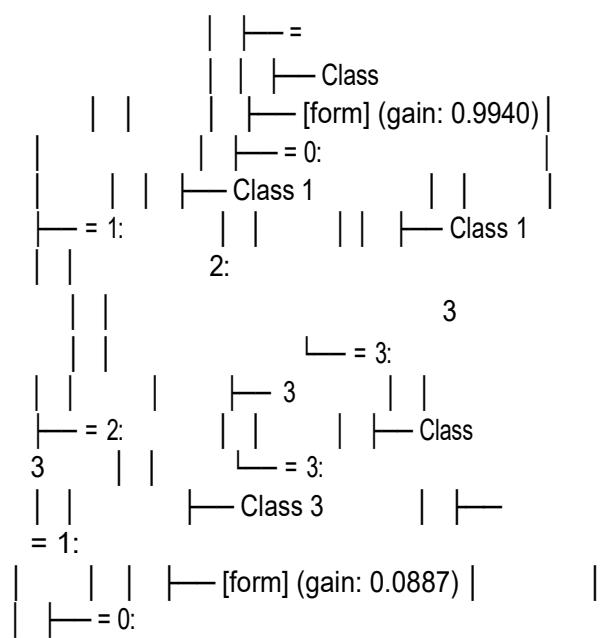
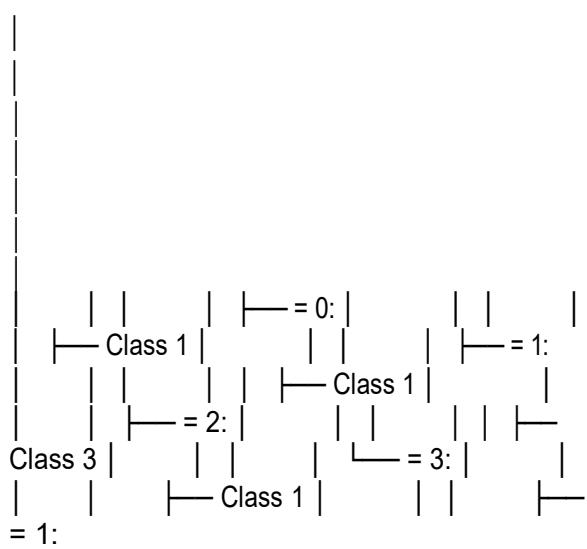


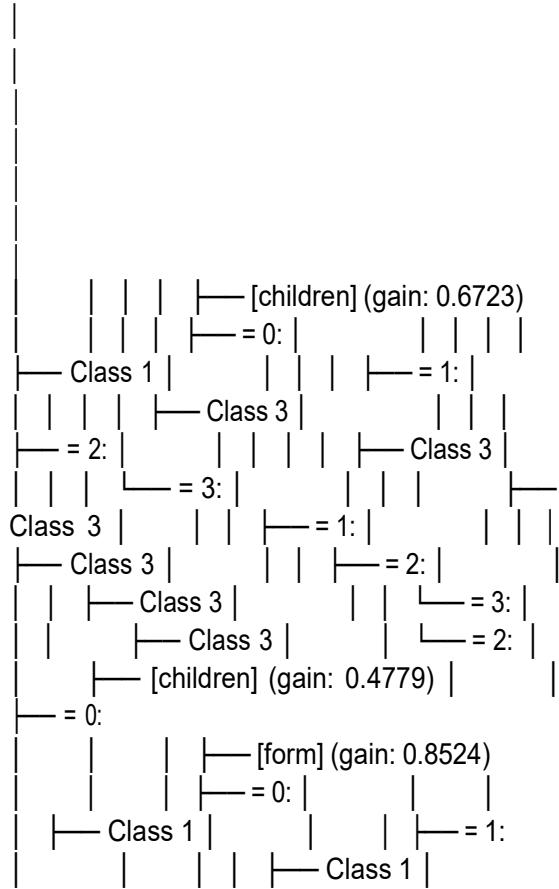




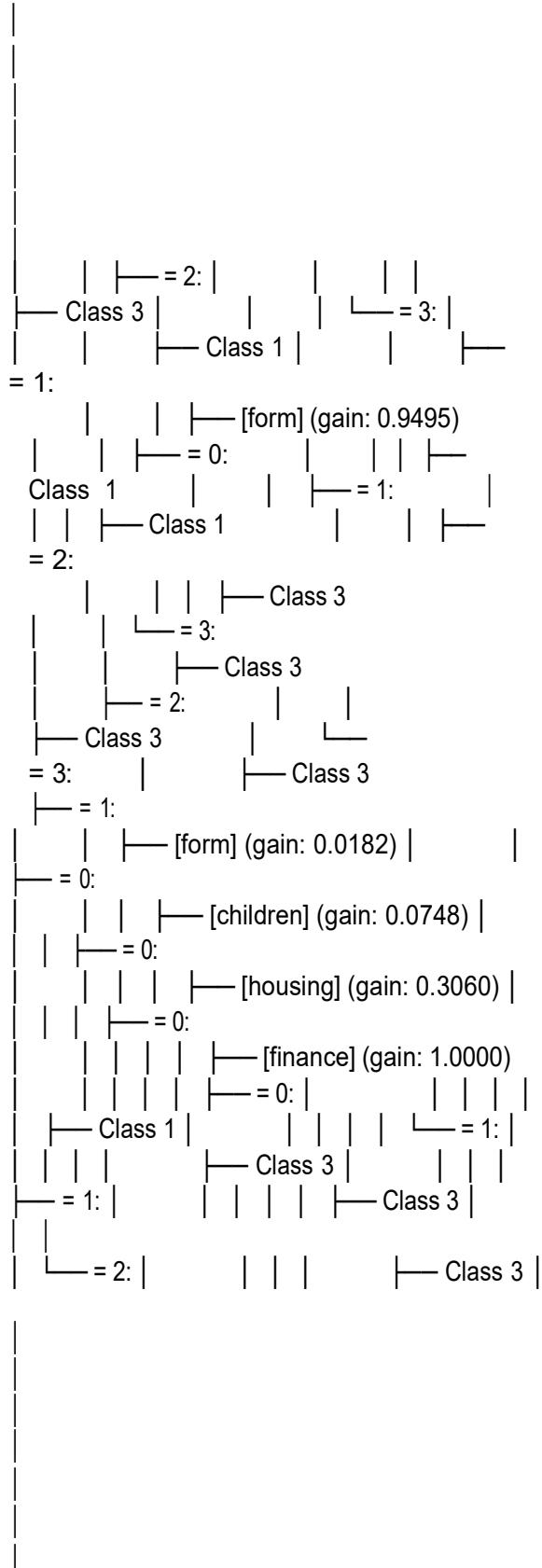


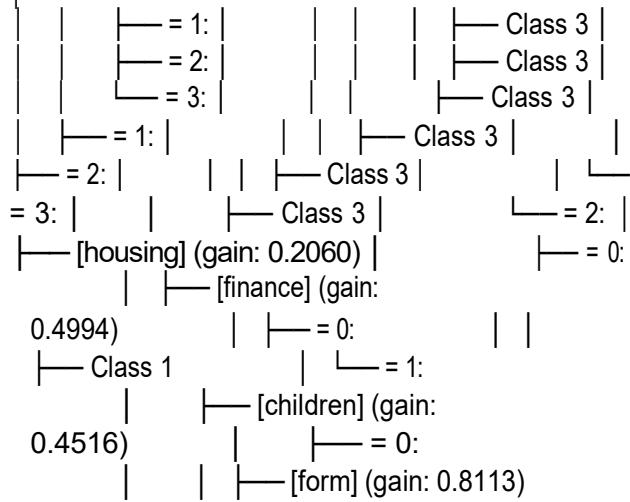






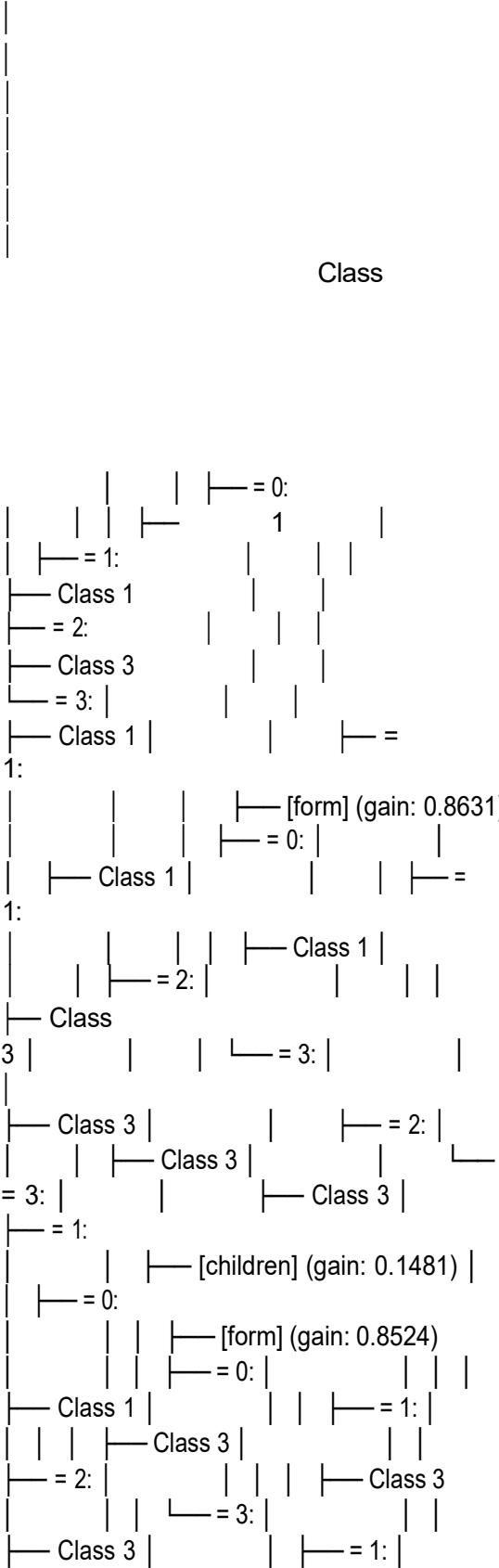
Class

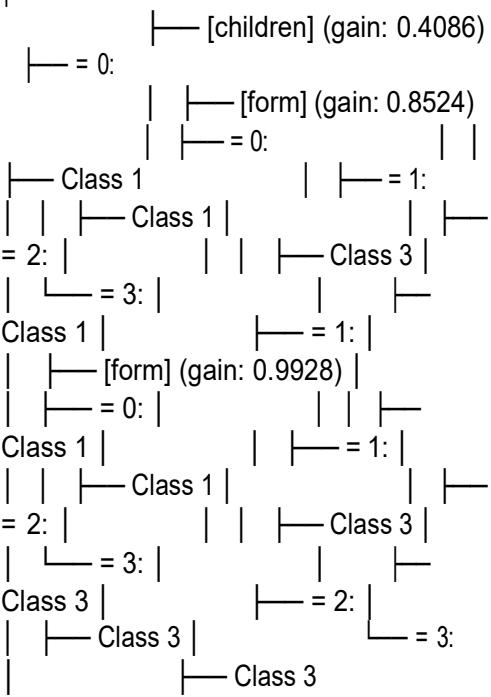
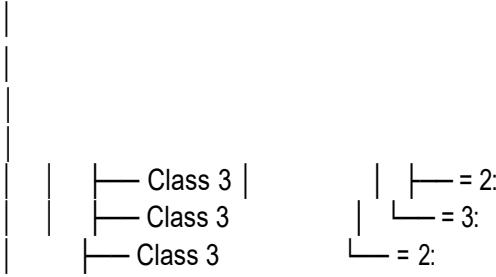




-----

-----





-----