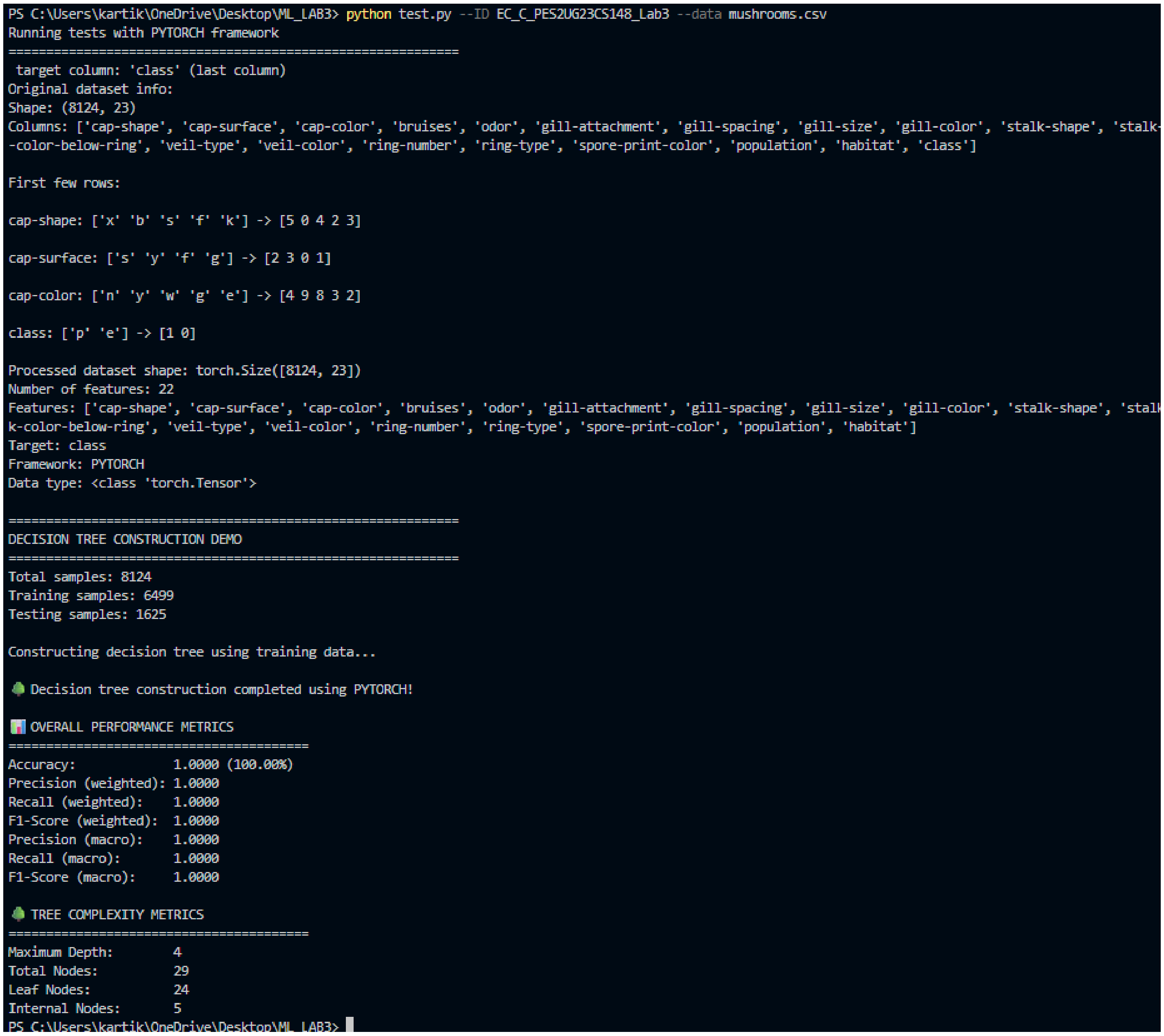
MACHINE LEARNING

DECISION TREE CLASSIFIER MULTI-DATASET ANALYSIS

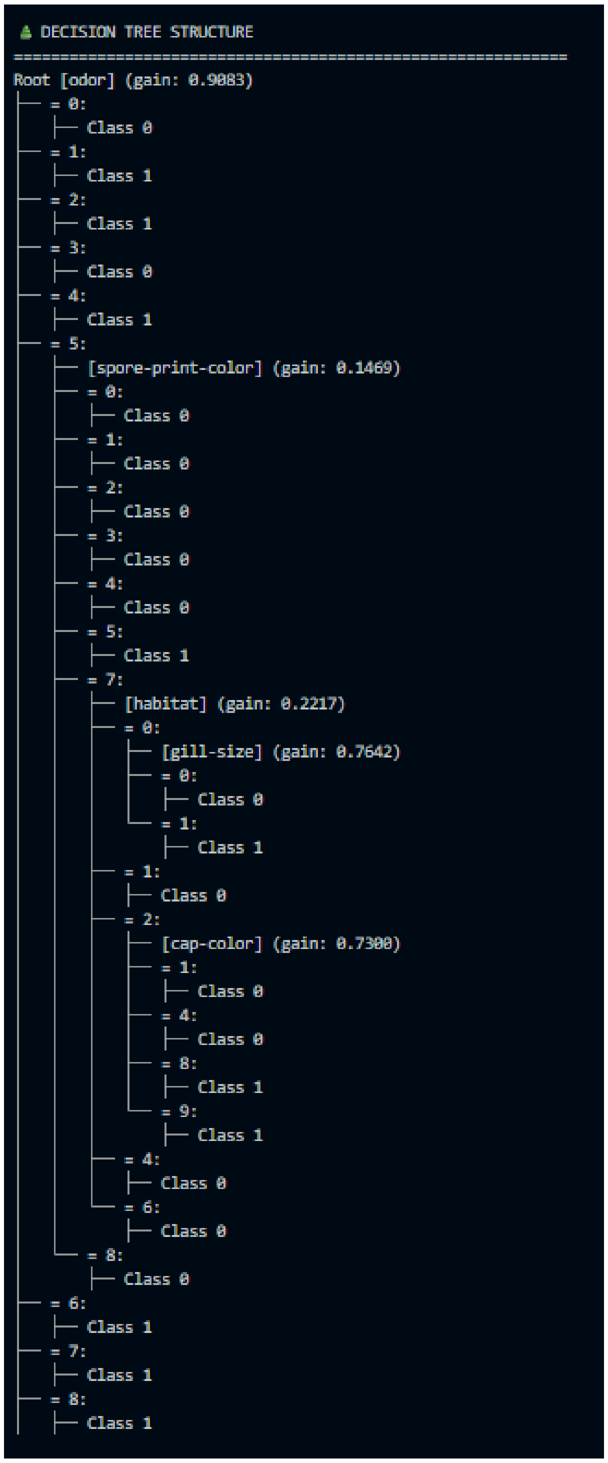
```
NAME: CHERUKURI VENKATA KARTIK
SEC:C
SRN:PES2UG23CS148
```

---

1. MUSHROOMS.CSV



```
PS C:\Users\kartik\OneDrive\Desktop\ML_LAB3> python test.py --ID EC_C_PES2UG23CS148_Lab3 --data mushrooms.csv
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-
-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', '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
k-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', '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!

📊 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\kartik\OneDrive\Desktop\ML_LAB3>
```

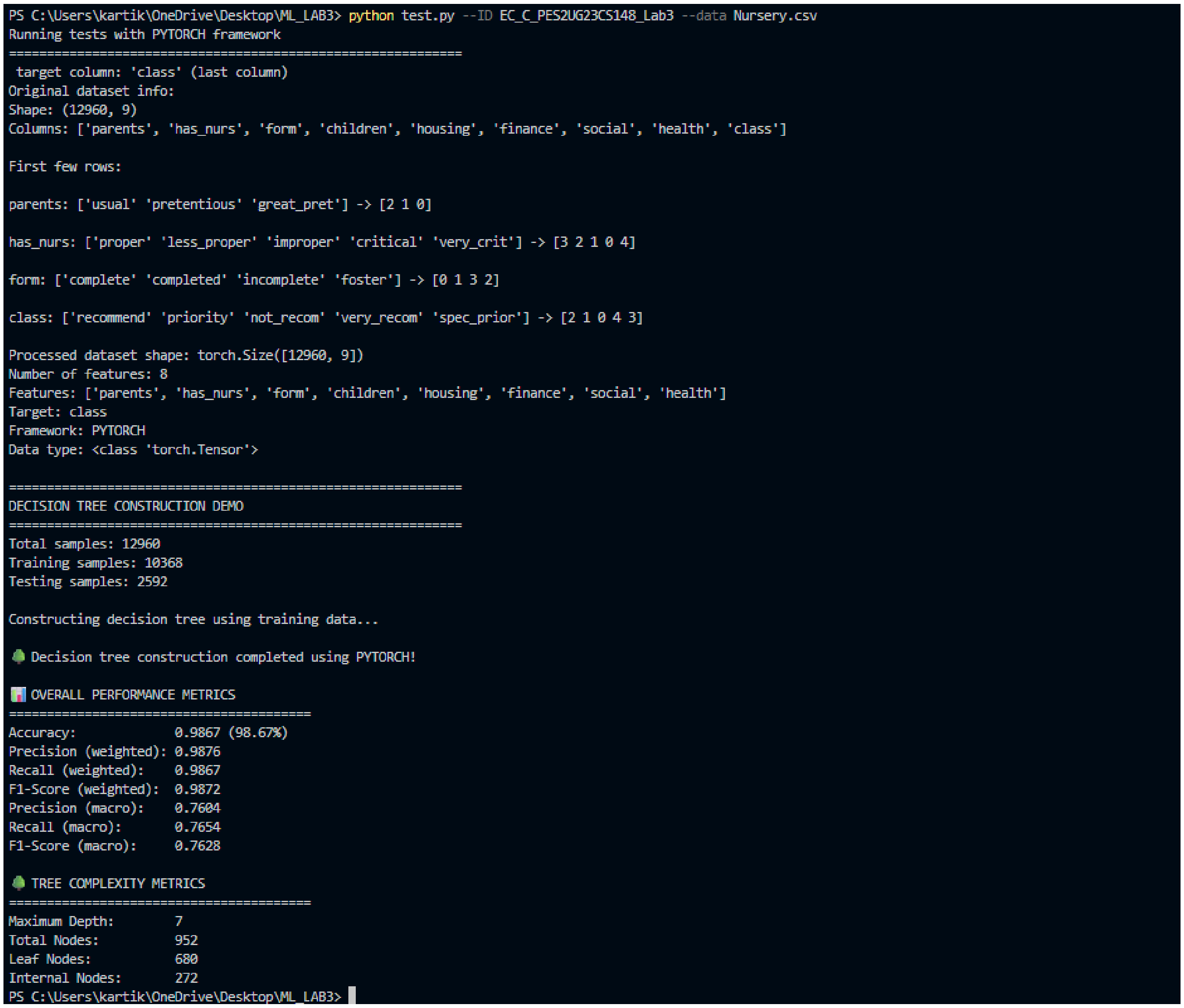


```
▲ DECISION TREE STRUCTURE
============================================================
Root [odor] (gain: 0.9083)
├── = 0:
│   ├── Class 0
├── = 1:
│   ├── Class 1
├── = 2:
│   ├── Class 1
├── = 3:
│   ├── Class 0
├── = 4:
│   ├── Class 1
├── = 5:
│   ├── [spore-print-color] (gain: 0.1469)
│   ├── = 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
```

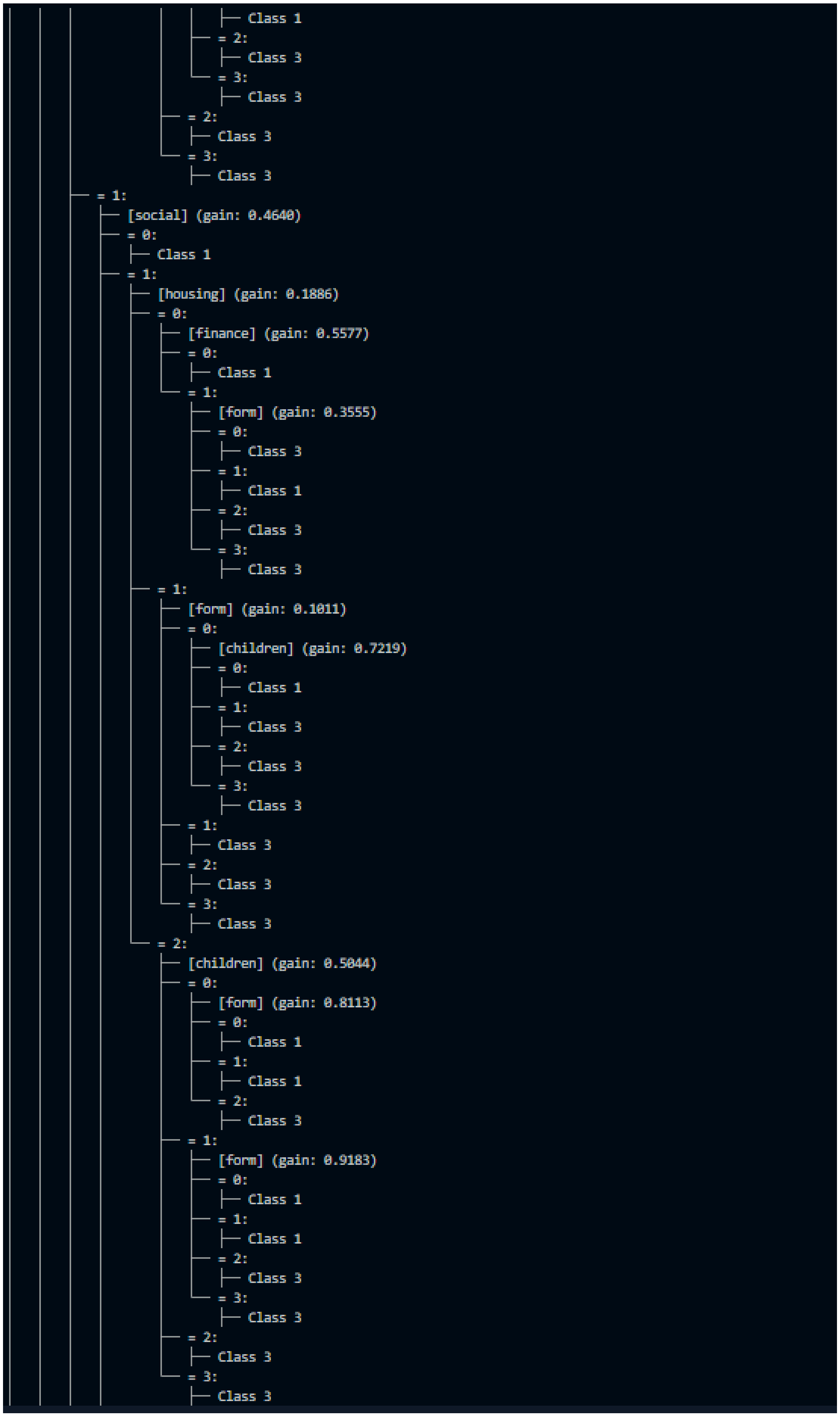
- Accuracy obtained is 100%
    - 100% accuracy tells us that the dataset is clean and perfectly-fit
- Precision ,Recall ,F1-Score (Weighted and Macro : 1.000 (100%))
    - Precision of 1.000 refers that 100% of the predictions of the model were correct
    - Recall of 1.000 refers that the model found all poisonous mushrooms i.e there were no misses
    - F1 score gives us the harmonic mean of precision and recall hence 1.000
- Maximum depth - 4 (Height of the tree)

- Total number of nodes-29
  - Leaf nodes - 24
  - Internal Nodes-5
- Root node of the tree is 'odor' then to 'spore-print-color' and 'habitat'
- It is a shallow tree because depth is 4 ,hence complexity is low
- 'Odor' contributes most to classification'
- Class Distribution: Balanced
- There is no overfitting

2.NURSERY.CSV



```
PS C:\Users\kartik\OneDrive\Desktop\ML_LAB3> python test.py --ID EC_C_PES2UG23CS148_Lab3 --data Nursery.csv
Running tests with PYTORCH framework
============================================================
 target column: 'class' (last column)
Original dataset info:
Shape: (12960, 9)
Columns: ['parents', 'has_nurs', 'form', 'children', 'housing', 'finance', 'social', 'health', 'class']

First few rows:

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...

🌳 Decision tree construction completed using PYTORCH!

📊 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\kartik\OneDrive\Desktop\ML_LAB3>
```

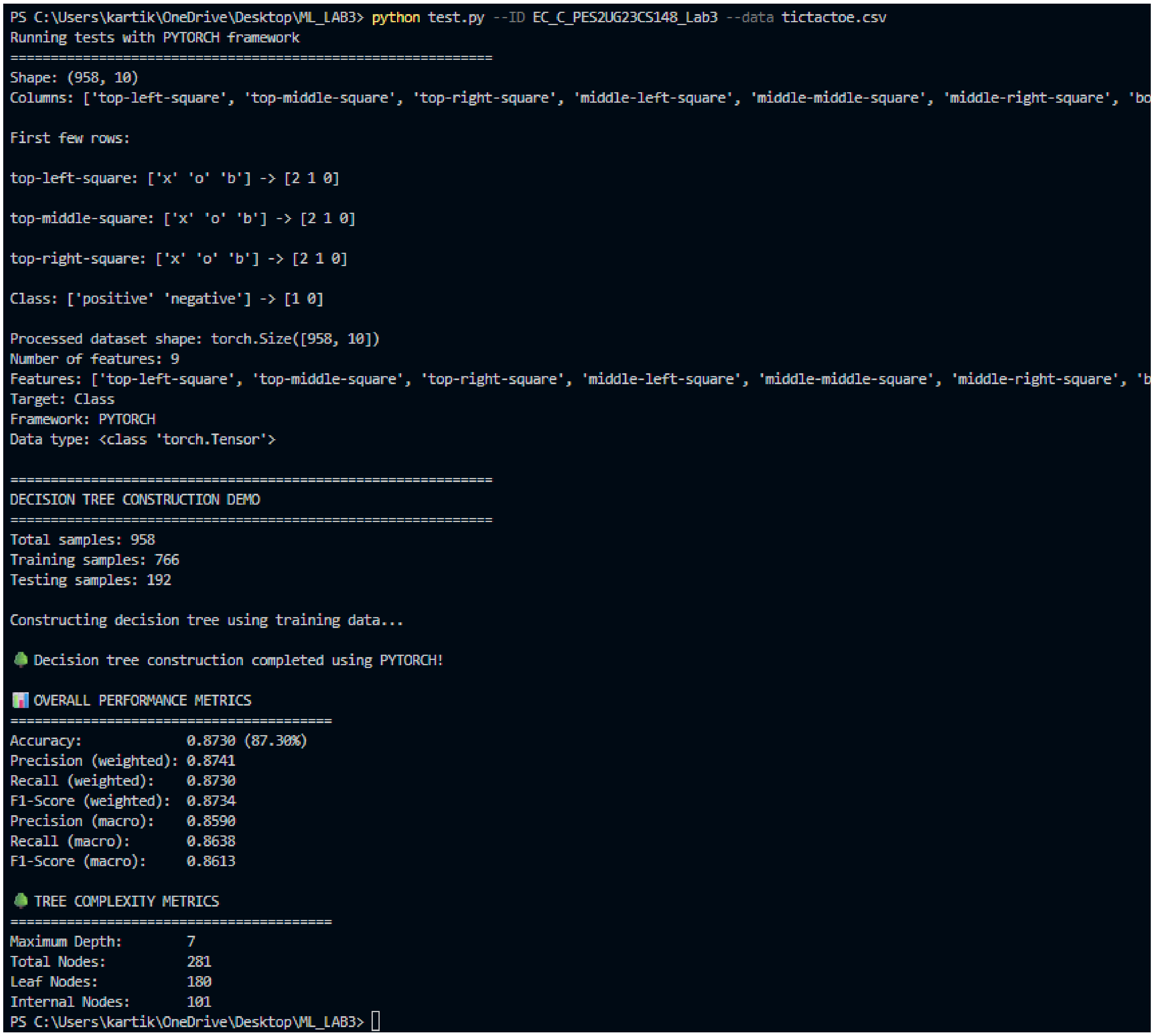


```
│   │   │           │   │   ├── Class 1
│   │   │           │   ├── = 2:
│   │   │           │   │   ├── Class 3
│   │   │           │   └── = 3:
│   │   │           │       ├── Class 3
│   │   │           ├── = 2:
│   │   │           │   ├── Class 3
│   │   │           └── = 3:
│   │   │               ├── 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:
│   │   │   │       │   ├── Class 3
│   │   │   │       └── = 3:
│   │   │   │           ├── Class 3
```

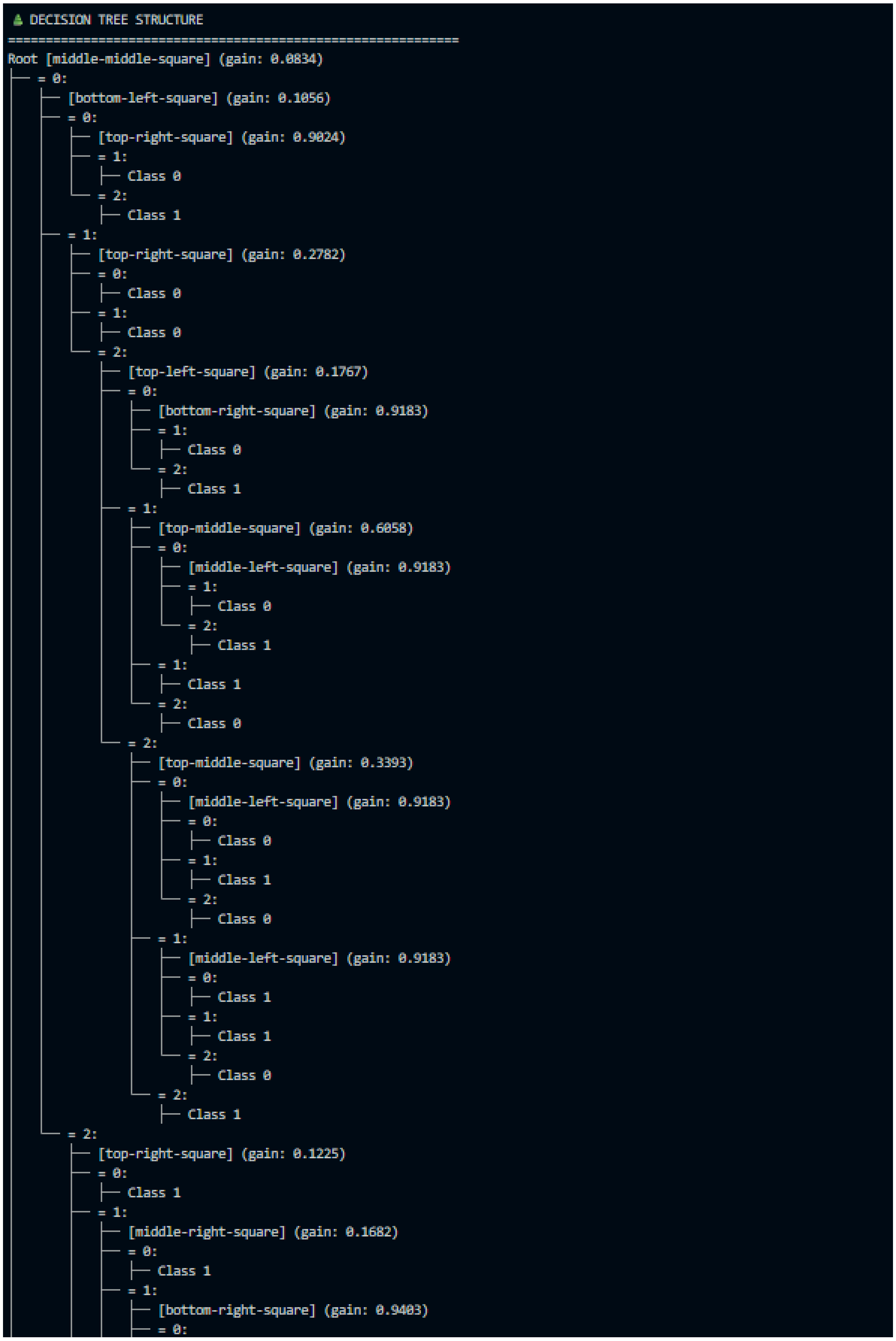
- Accuracy obtained is 98.67%

  - 98.67% accuracy tells us that the dataset is almost clean with very less noise
- Precision (Weighted:0.9876 , Macro:0.7604)
  - Precision of 0.9876 refers that most of the predictions made by the model were correct
  - Macro score of 0.7604 tells us that minority classes have lower precision
- Recall (Weighted:0.9867, Macro:0.7654)
  - Recall of 0.9867 refers that the model found almost all of the true positives
  - Macro score of 0.7654 tells us that model may find it difficult to detect minority classes
- F1-Score (Weighted:0.9872 and Macro :0.7628)
  - Weighted F1 score is high , hence balanced performance
  - Macro F1 is is comparatively lower hence performance might not be balanced
- Maximum depth - 7 (Height of the tree)
- Total number of nodes-952
  - Leaf nodes - 680
  - Internal Nodes-272
- Early splits typically on finance / social / health
- It is a very big tree because depth is 7 and there are many leaf and internal nodes ,hence complexity is high
- 'Finance' ,'Social','Health'contributes most to classification'
- Class Distribution: Unbalanced
- There is Overfitting

3.TICTACTOE.CSV



```
PS C:\Users\kartik\OneDrive\Desktop\ML_LAB3> python test.py --ID EC_C_PES2UG23CS148_Lab3 --data tictactoe.csv
Running tests with PYTORCH framework
============================================================
Shape: (958, 10)
Columns: ['top-left-square', 'top-middle-square', 'top-right-square', 'middle-left-square', 'middle-middle-square', 'middle-right-square', 'bo

First few rows:

top-left-square: ['x' 'o' 'b'] -> [2 1 0]

top-middle-square: ['x' 'o' 'b'] -> [2 1 0]

top-right-square: ['x' 'o' 'b'] -> [2 1 0]

Class: ['positive' 'negative'] -> [1 0]

Processed dataset shape: torch.Size([958, 10])
Number of features: 9
Features: ['top-left-square', 'top-middle-square', 'top-right-square', 'middle-left-square', 'middle-middle-square', 'middle-right-square', 'b
Target: Class
Framework: PYTORCH
Data type: <class 'torch.Tensor'>

============================================================
DECISION TREE CONSTRUCTION DEMO
============================================================
Total samples: 958
Training samples: 766
Testing samples: 192

Constructing decision tree using training data...

🌳 Decision tree construction completed using PYTORCH!

📊 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
PS C:\Users\kartik\OneDrive\Desktop\ML_LAB3> 
```



```
🌲 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:
│   │           ├── [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:
```

- Accuracy obtained is 87.30%

    - 87.30% accuracy tells us that the dataset is not that clean i.e accuracy can be improved
- Precision (Weighted:0.8741 , Macro:0.8590)
    - Precision of 0.8741 refers that most of the predictions made by the model were correct but lesser than the previous datasets
    - Macro score of 0.8590 tells us that minority classes may have slightly lower precision
- Recall (Weighted:0.8730, Macro:0.8638)
    - Recall of 0.8730 refers that the model found most of the true positives
    - Macro score of 0.8638 tells us that model is better at finding true positives across all classes
- F1-Score (Weighted:0.8734 and Macro :0.8613)
    - Weighted F1 score is high , hence balanced performance
    - Macro F1 is almost equal hence performance might tend towards balanced
- Maximum depth - 7 (Height of the tree)
- Total number of nodes-281
    - Leaf nodes - 180
    - Internal Nodes-101
- Early splits typically on midlle-middle-square
- It is a big tree because depth is 7 but leaf nodes are lesser than previous tree so complexity is medium
- 'Centre-Square' contributes most to classification'
- Class Distribution: Slightly Unbalanced
- There might be Overfitting

Q4
(a)**Algorithm Performance**

- (a)Mushrooms has the highest accuracy of 100% because there is very less or no noise
- (b)Larger dataset increases the number of test cases and hence accuracy improves , smaller datasets may result in lower accuracy
- (c)Features help if they can distinguish between classes

(b)**Data Characteristics Impact**

- Class imbalance decreases the overall F1 score, for example in nursery weighted values show good scores whereas macro scores are poor
- Multi-valued features work better

(c)**Real world scenarios**

- Mushrooms ->Food Safety
- Nursery ->University Student Record
- Tic-Tac-Toe ->Network Security