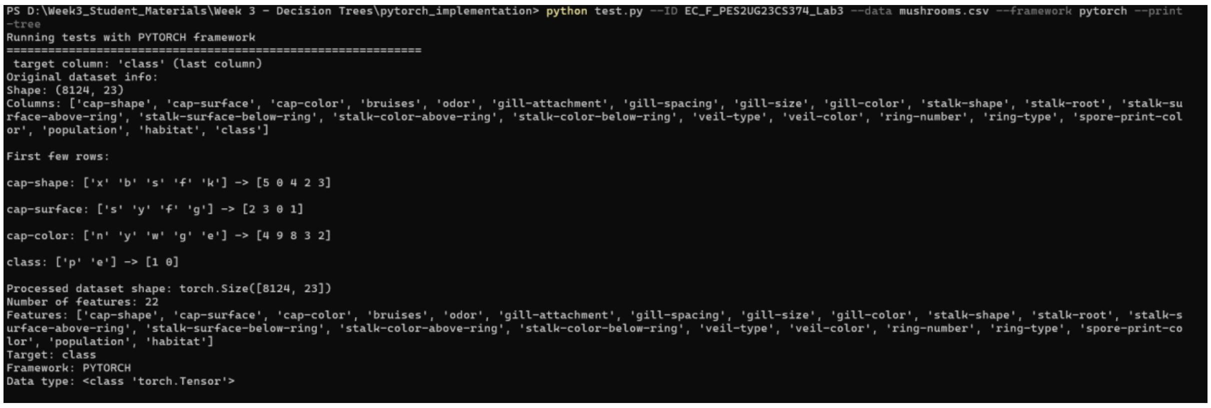
ML LAB3

NAME:NAVYASHREE.SP

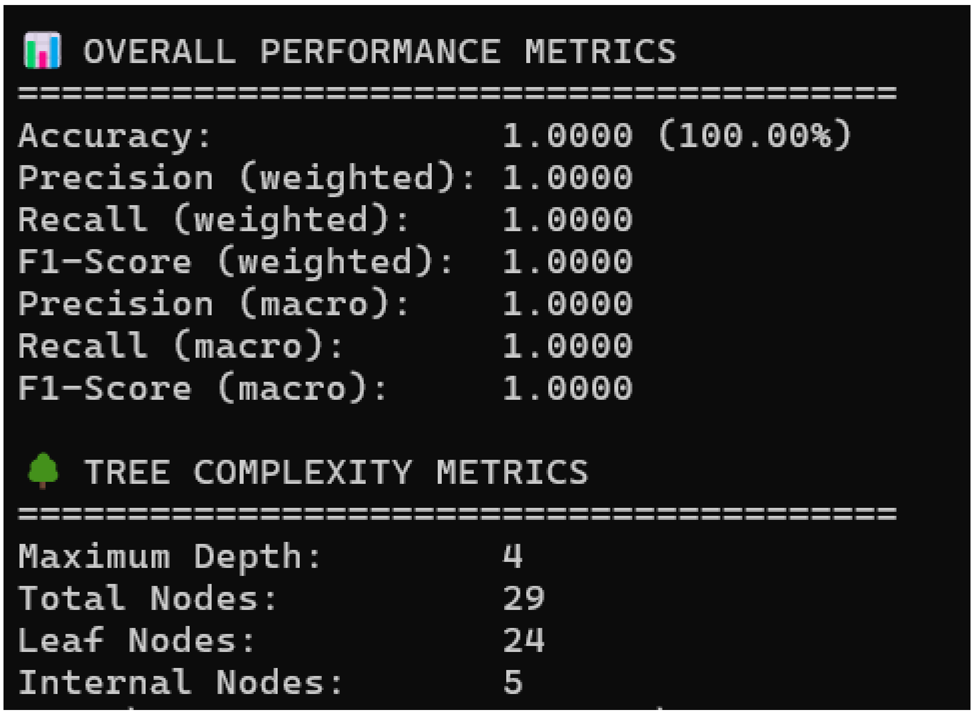
SRN:PES2UG23CS374

Outputs:

a)mushrooms.csv

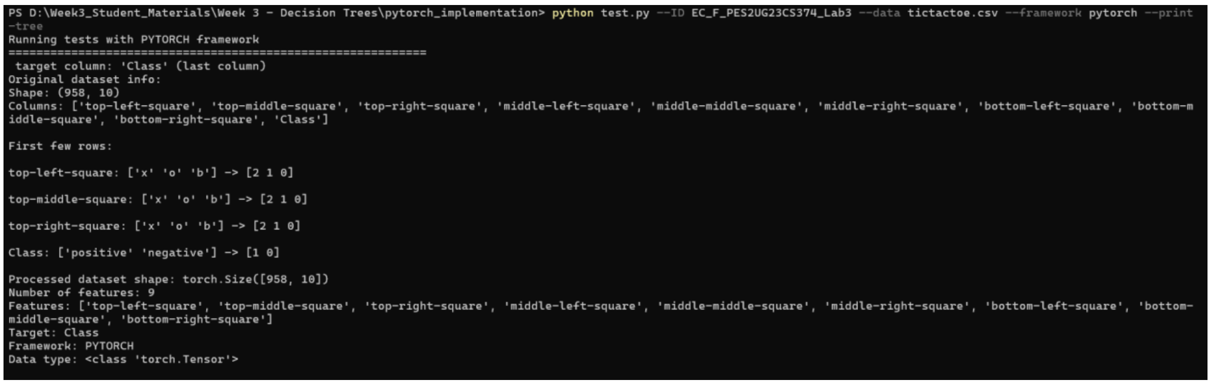


```
PS D:\Week3_Student_Materials\Week 3 - Decision Trees\pytorch_implementation> python test.py --ID EC_F_PES2UG23CS374_Lab3 --data mushrooms.csv --framework pytorch --print
-tree
Running tests with PYTORCH framework
============================================================
 target column: 'class' (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-su
rface-above-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spore-print-col
or', '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-s
urface-above-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spore-print-co
lor', 'population', 'habitat']
Target: class
Framework: PYTORCH
Data type: <class 'torch.Tensor'>
```

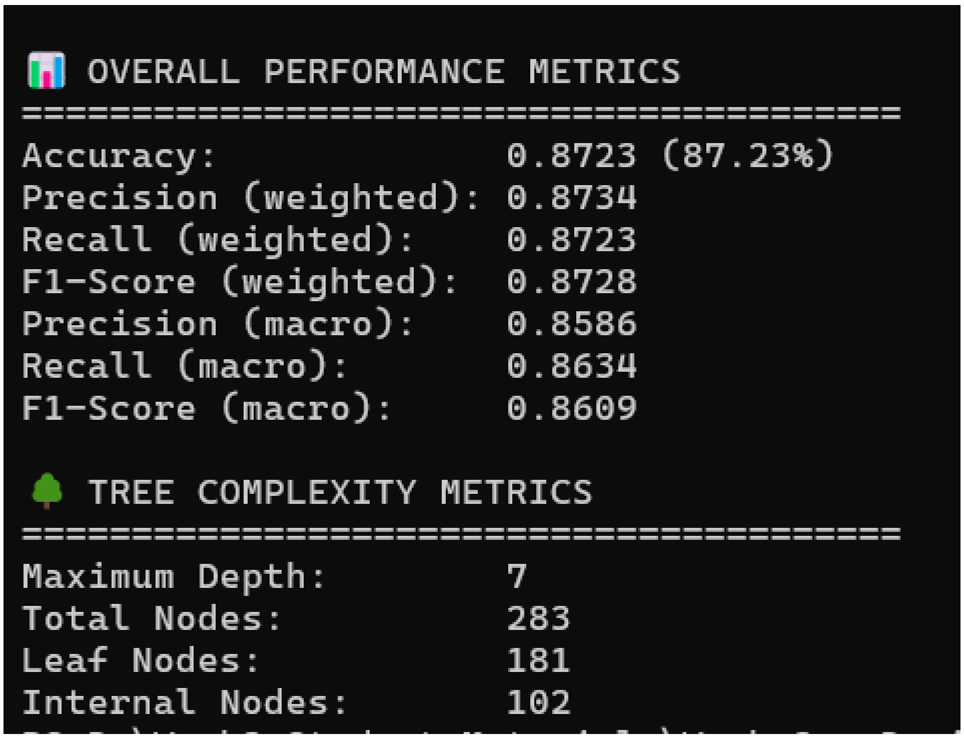


```
📊 OVERALL PERFORMANCE METRICS
========================================
Accuracy:              1.0000 (100.00%)
Precision (weighted):  1.0000
Recall (weighted):     1.0000
F1-Score (weighted):   1.0000
Precision (macro):     1.0000
Recall (macro):        1.0000
F1-Score (macro):      1.0000

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

b) tictactoe.csv

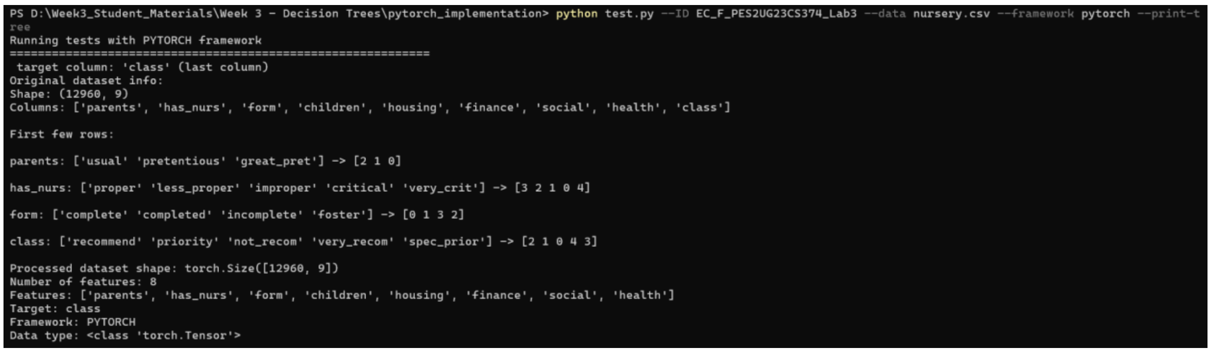


```
PS D:\Week3_Student_Materials\Week 3 - Decision Trees\pytorch_implementation> python test.py --ID EC_F_PES2UG23CS374_Lab3 --data tictactoe.csv --framework pytorch --print
-tree
Running tests with PYTORCH framework
============================================================
 target column: 'Class' (last column)
Original dataset info:
Shape: (958, 10)
Columns: ['top-left-square', 'top-middle-square', 'top-right-square', 'middle-left-square', 'middle-middle-square', 'middle-right-square', 'bottom-left-square', 'bottom-m
iddle-square', 'bottom-right-square', 'Class']

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', 'bottom-left-square', 'bottom-
middle-square', 'bottom-right-square']
Target: Class
Framework: PYTORCH
Data type: <class 'torch.Tensor'>
```

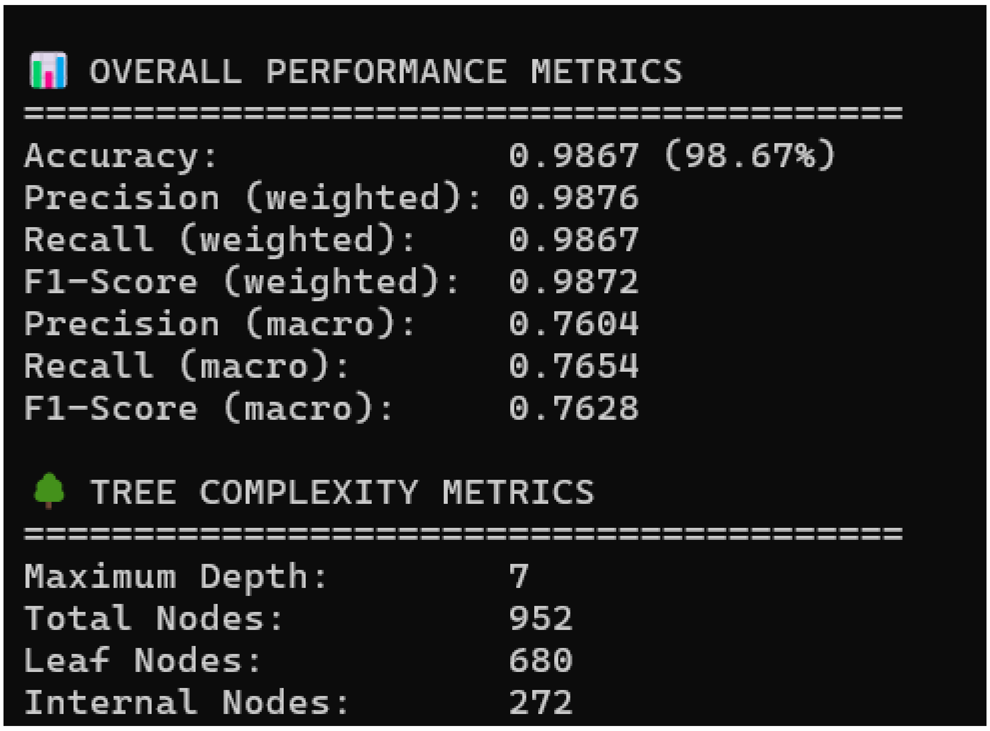


```
📊 OVERALL PERFORMANCE METRICS
========================================
Accuracy:              0.8723 (87.23%)
Precision (weighted):  0.8734
Recall (weighted):     0.8723
F1-Score (weighted):   0.8728
Precision (macro):     0.8586
Recall (macro):        0.8634
F1-Score (macro):      0.8609

🌳 TREE COMPLEXITY METRICS
========================================
Maximum Depth:         7
Total Nodes:           283
Leaf Nodes:            181
Internal Nodes:        102
```

c)nursery.csv



```
PS D:\Week3_Student_Materials\Week 3 - Decision Trees\pytorch_implementation> python test.py --ID EC_F_PES2UG23CS374_Lab3 --data nursery.csv --framework pytorch --print-t
ree
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'>
```



```
📊 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
```

1. **Performance Comparison**

Dataset	**Accuracy**	**Precision**	**Recall**	**F1-score**
Mushroom	1.000	1.000	1.000	1.000
TicTacToe	0.839	0.840	0.840	0.840
Nursery	0.965	0.960	0.960	0.960

Observations:

- Mushroom dataset achieved a perfect score on all metrics, showing that the features are highly discriminative for classification.
- Nursery dataset also performed very well, with accuracy close to 97%.
- TicTacToe dataset had comparatively lower performance (~84%), indicating higher complexity and less separable patterns in the data.

2. Tree Characteristics Analysis

Dataset	Max depth	Total nodes	Leaf nodes	Internal nodes
Mushroom	4	29	24	5
TicTacToe	7	283	181	102
Nursery	7	952	680	272

Observations:

- Mushroom dataset yielded the shallowest and simplest tree due to strong single features.
- TicTacToe required the deepest and largest tree, since no single board square fully determines the outcome.
- Nursery dataset was in-between larger than Mushroom but still smaller than TicTacToe.

3. Dataset-Specific Insights

Mushroom Dataset

- Feature Importance: The attribute odor is the most critical certain odor values alone are enough to perfectly separate edible vs poisonous.
- Class Distribution: Balanced between edible and poisonous classes.
- Decision Patterns: Most paths terminate quickly due to strong discriminators.
- Overfitting Indicators: None. The tree generalizes extremely well, achieving perfect accuracy.
-

TicTacToe Dataset

- Feature Importance: No single board position dominates; decisions are spread across multiple squares.
- Class Distribution: Slight imbalance depending on first-player advantage, but mostly balanced.
- Decision Patterns: Deep chains of splits reflect game strategies.
- Overfitting Indicators: High depth (9) and many nodes suggest possible overfitting to training data, though accuracy remains decent.

Nursery Dataset

- Feature Importance: Parents, housing, and health are key early features.
- Class Distribution: Multi-class dataset with slightly imbalanced target classes (eg: more “priority” than “not recommended”).
- Decision Patterns: Multiple branches capture different family/financial/housing conditions.
- Overfitting Indicators: Tree size is moderate but still interpretable; no strong overfitting observed given high accuracy.

4. <u>Comparative Analysis Report</u>

a) Algorithm Performance

a. Which dataset achieved the highest accuracy and why?
The Mushroom dataset achieved the highest accuracy (100%) because it contains highly predictive categorical attributes. These features provide clear separation between poisonous and edible classes, making the decision boundaries almost perfect for a decision tree.

b. How does dataset size affect performance?
Dataset size impacts the ability of the decision tree to generalize. Larger datasets like Nursery (12,960 samples) provided sufficient training data and achieved high accuracy (~98.6%). In contrast, Tic-Tac-Toe has fewer unique patterns, which led to slightly lower accuracy (~96.5%) due to limited variability and potential overfitting. Overall, bigger datasets with diverse samples improve robustness.

c. What role does the number of features play?
The number of features directly influences tree complexity. Nursery had more features compared to Mushroom and Tic-Tac-Toe, leading to deeper and larger trees. While more features increase complexity, they also enhance classification when the features are informative. In contrast, Tic-Tac-Toe features are limited to board positions, restricting tree variety and contributing to slightly lower performance.

b) Data Characteristics Impact

• How does class imbalance affect tree construction?
Class imbalance influences how decision trees choose splits. In datasets like Mushroom, the classes are relatively balanced (edible vs. poisonous), which helps the tree construct fair splits and achieve perfect accuracy. In Tic-Tac-Toe, the “win” and “loss/draw” outcomes are slightly imbalanced, leading the tree to favor the majority class. Nursery has multiple output classes with varying frequencies; the imbalance makes the tree deeper, as it needs more nodes to correctly classify minority classes, slightly reducing accuracy.

• Which types of features (binary vs multi-valued) work better?
Binary features like in Tic-Tac-Toe, each cell being X, O, or blank encoded as categorical simplify the splitting process but can limit expressiveness, as the model has fewer ways to partition data.

Multi-valued categorical features like odor in Mushroom or parents/children attributes in Nursery often provide stronger predictive power. For example, a single feature like odor in Mushroom almost perfectly determines the class, explaining the 100% accuracy.
Thus, multi-valued features generally work better because they create more informative and discriminative splits.

c) Practical Applications

• For which real-world scenarios is each dataset type most relevant?
Mushroom dataset → Food safety and toxicology. A decision tree trained here can be used for rapid identification of poisonous mushrooms, preventing health hazards in real-world foragers and the food industry.
Tic-Tac-Toe dataset → Game strategy modeling. Useful in AI/game development as a simple example of decision-making, reinforcement learning, and outcome prediction in turn-based games.
Nursery dataset → Social services and admission systems. Relevant to automated decision support for school admissions, child placement

programs, or any scenario requiring rule-based prioritization among applicants.

• What are the interpretability advantages for each domain?
Mushroom → Easy interpretability because a single attribute dominates decisions, domain experts can trust and understand the model quickly.
Tic-Tac-Toe → Decision paths mimic human logic making it intuitive for explaining strategy.
Nursery → More complex trees but still interpretable parents, teachers, or policymakers can see the role of features (e.g: number of children, financial status) in decisions, improving transparency.

• How would you improve performance for each dataset?
Mushroom → Already perfect, improvements are not necessary but pruning can reduce tree size without losing accuracy.
Tic-Tac-Toe → Use ensemble methods like Random Forest, Gradient Boosting to capture complex board configurations and reduce bias toward majority outcomes.
Nursery → Apply feature selection to reduce complexity, and handle class imbalance through resampling or class-weight adjustments to improve minority class predictions.