DHRUV SUDHAN NAIK PES2UG23CS174 5C ML LAB3

```
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-b
elow-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-root', 'stalk-surface-above-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-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
```

```
@E7VN ➜ /workspaces/ML-Lab-3 (main) $ python test.py --ID EC_SC_PES2UG23CS174_Lab3 --data tictactoe.csv
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-middle-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'>

============================================================
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
@E7VN ➜ /workspaces/ML-Lab-3 (main) $
```

```
• @E7VN →/workspaces/ML-Lab-3 (main) $ python test.py --ID EC_5C_PES2UG23CS174_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
```

Characteristic	Mushroom	Tic-Tac-Toe	Nursery
Max Depth	4	7	7
Total Nodes	29	281	952
Leaf Nodes	24	180	680
Internal Nodes	5	101	272
Node:Sample Ratio	1:280	1:3.4	1:13.6

Aspect	Mushroom	Tic-Tac-Toe	Nursery
Feature Type	Biological	Spatial	Socio-economic
Class Balance	Perfect	Moderate	Severe imbalance
Tree Simplicity	Very Simple	Moderate	Very Complex
Overfitting Risk	None	Low	High
Decision Clarity	Clear rules	Strategic patterns	Complex policies

a) Which dataset achieved the highest accuracy and why?

The Mushroom dataset achieved perfect classification with 100% accuracy because:

- Clear Biological Rules: Physical traits such as odor, spore-print-color, and gill-color have direct and definitive relationships with edibility.

- Perfect Feature Discriminability: Certain attributes—for example, "odor=foul" always indicating poisonous and "odor=almond" always indicating edible—provide unambiguous cues.
- No Ambiguity: The biological characteristics create deterministic classification rules without exceptions.
- High-Quality Features: All 22 features are highly relevant and non-redundant, enhancing model clarity and precision.

b) How does dataset size affect performance?

Dataset size influences performance less than data quality:

- The Mushroom dataset, with 8,124 samples, achieved flawless accuracy, demonstrating the importance of quality over quantity.
- The Nursery dataset, larger at 12,960 samples, reached 98.67% accuracy but was affected by class imbalance, which impacted the performance on minority classes.
- Tic-Tac-Toe (958 samples): 87.30% - smallest but good performance due to clear patterns - Hence, Data quality and feature relevance matter more than sheer volume

Q) What role does the number of features play?

- The number of features is less critical than their quality:
    - **Mushroom (22 features):** Produced the simplest decision tree (29 nodes) since all features are highly relevant.
    - **Tic-Tac-Toe (9 features):** Resulted in a medium-complexity tree (281 nodes) with meaningful spatial features.
    - **Nursery (8 features):** Created the most complex tree (952 nodes) due to complex interactions among social features.
- Therefore, having highly relevant features reduces tree complexity more effectively than simply having more features.

b) Howdoes class imbalance affect tree construction?

- Class imbalance severely impacts performance on minority classes:
    - **Mushroom (Balanced):** Achieved perfect results with 100% across all metrics.

- **Tic-Tac-Toe (Moderate imbalance):** Showed a small 3% difference between macro- and weighted-scores, an acceptable gap.
- **Nursery (Severe imbalance):** Suffered a large 22% gap, indicating poor minority class performance.

- Imbalances cause trees to bias toward majority classes, often neglecting rare class patterns.
- For example, the Nursery dataset's macro F1 score (76%) is much lower than its weighted F1 score (98%), highlighting minority class neglect.

Q) Which types of features (binary vs. multi-valued) work better?

- Both binary and multi-valued features can be effective depending on context:
    - **Binary Features (Tic-Tac-Toe):** Works well for spatial decisions (X/O/blank).
    - **Multi-valued Features (Mushroom):** Multiple categories (e.g., 6 odors, 9 colors) provide rich discriminative information.
    - **Mixed Types (Nursery):** Combination requires careful handling.
- Ultimately, feature discriminability matters more than the number of distinct values.

c) Howto improve performance for each dataset:

- **Mushroom (Already Perfect):**
    - Include more mushroom species variations.
    - Add geographical and seasonal data.
    - Enhance with chemical composition features.
- **Tic-Tac-Toe (87.30% → 90%+):**
    - Incorporate move sequence history instead of just static board state.
    - Add player skill level as a feature.

- Use pruning techniques to reduce overfitting.

- **Nursery (98.67% weighted → 85%+ macro):**
  - Address class imbalance using techniques like SMOTE oversampling or class weighting.
  - Perform feature engineering to create composite socio-economic indices.
  - Apply tree pruning to reduce complexity from 952 nodes and prevent overfitting.
  - Use cost-sensitive learning to prioritize accuracy on minority classes.