MACHINE LEARNING

LAB 3

Name: Diya Prakash **SRN**: PES2UG23CS184

Section: 5C CSE

Output Screenshots:

1) Mushrooms Dataset:

```
PS C:\Users\diyap\OneDrive\Documents\sem5\ML\all\code\pytorch_implementation> python test.py --ID EC_C_PES2UG23CS184_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-root', 'stalk-surface-above-ring', 'stalk-surfa
ce-below-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: ['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-surf
ace-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
PS C:\Users\diyap\OneDrive\Documents\sem5\ML\all\code\pytorch_implementation>
```

2) Tictactoe Dataset:

```
PS C:\Users\diyap\OneDrive\Documents\sem5\ML\all\code\pytorch_implementation> python test.py --ID EC_C_PES2UG23CS184_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-sq
uare', '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-s
quare']
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...
```

```
📊 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\diyap\OneDrive\Documents\sem5\ML\all\code\pytorch_implementation>
```

3) Nursery Dataset:

```
PS C:\Users\diyap\OneDrive\Documents\sem5\ML\all\code\pytorch_implementation> python test.py --ID EC_C_PES2UG23CS184_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\diyap\OneDrive\Documents\sem5\ML\all\code\pytorch_implementation>
```

Note: Although the --print-tree option was available for visualizing the complete decision trees, including the full text output would occupy excessive space; hence, only the performance outputs are presented here, while the tree structure has been analyzed separately for the discussion.

1. Performance Comparison

Dataset	Accuracy	Precision (weighted)	Recall (weighted)	F1-Score (weighted)
Mushrooms	1.0000	1.0000	1.0000	1.0000
Tictactoe	0.8730	0.8741	0.8730	0.8734
Nursery	0.9867	0.9876	0.9867	0.9872

2. Tree Characteristics Analysis

Dataset	Max Depth	Total Nodes	Leaf Nodes	Internal Nodes	Most Important Features	Tree Complexity Insights
Mushrooms	4	29	24	5	odor, spore-print-color, habitat	Very shallow and simple tree due to highly informative root feature.
Tictactoe	7	281	180	101	middle-middle-square, bottom-left-square, top-right-square, bottom-right-square	Moderate complexity; many features contribute, reflecting game logic.
Nursery	7	952	680	272	social, form, housing	Very large and deep tree, reflecting multi-class, multi-valued features.

3. Dataset-Specific Insights

Mushrooms

- **Feature Importance:** 'odor' is the dominant feature; most splits are decided at the root.
- **Class Distribution:** Perfect separation; likely balanced or easily separable classes.
- **Decision Patterns:** Most samples classified at the root; few deeper splits.
- **Overfitting Indicators:** None; tree is small and generalizes perfectly.

Tictactoe

- **Feature Importance:** 'middle-middle-square' is most important, followed by other board positions.
- **Class Distribution:** Likely balanced (positive/negative).
- **Decision Patterns:** Multiple features needed; reflects complexity of game states.

- **Overfitting Indicators:** Moderate tree size; no severe overfitting.

Nursery

- **Feature Importance:** 'social', 'form', and 'housing' are key; many features used.
- **Class Distribution:** Multi-class, possibly imbalanced.
- **Decision Patterns:** Deep, complex paths; many features interact.
- **Overfitting Indicators:** Large tree size may indicate some overfitting, but high accuracy suggests good fit.

4. Comparative Analysis Report

a) Algorithm Performance

- **Highest Accuracy:** Mushrooms (100%)—due to a single highly informative feature ('odor') that almost perfectly separates classes.
- **Dataset Size Effect:** Larger datasets (mushrooms, nursery) do not reduce accuracy if features are informative; complexity increases with more classes/features.
- **Number of Features:** More features (nursery) lead to deeper, more complex trees; if one feature is dominant (mushrooms), tree remains shallow.

b) Data Characteristics Impact

- **Class Imbalance:** Can lead to deeper trees and more splits for minority classes (nursery).
- **Feature Types:** Binary features (tictactoe) lead to moderate complexity; multi-valued features (nursery) increase tree size and depth.
- **Best Performing Features:** Highly informative features (odor in mushrooms) simplify the tree and improve performance.

c) Practical Applications

- **Mushrooms:** Useful for food safety and toxicology; easy to interpret and deploy.
- **Tictactoe:** Game AI, pattern recognition; interpretable for strategy analysis.
- **Nursery:** Admission/classification systems; complex but interpretable for policy decisions.
- **Interpretability:** Mushrooms is most interpretable (shallow tree); nursery is most complex but still traceable.
- **Improvements:** For nursery, consider pruning or feature selection to reduce overfitting. For tictactoe, ensemble methods could improve accuracy. For mushrooms, model is already optimal.