NAME : CHARAN M REDDY

SEMESTER : 5th

SECTION : C

COURSE : MACHINE LEARNING

DATE : 19/08/2025

Screenshots of the 3 outputs for the 3 data sets

<u>mushroom.csv</u>

```
PS C:\Users\Chara\OneDrive\Desktop\PESU\ML\LABS\Lab2\all> python test.py --ID EC_C_PES2UG23CS146_Lab3 --data mushroom.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-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 've
pe', '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', 'v
ype', '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
```

tictactoe.csv

```
PS C:\Users\Chara\OneDrive\Desktop\PESU\ML\LABS\Lab2\all> python test.py --ID EC_C_PES2UG23CS146_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
```

Nursery.csv

```
PS C:\Users\Chara\OneDrive\Desktop\PESU\ML\LABS\Lab2\all> python test.py --ID EC_C_PES2UG23CS146_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
```

1) Performance Comparision

Datasets	Accuracy	Precision	Recall	F1-Score
mushroom	100%	1	1	1
tictactoe	87.3%	0.8741	0.8730	0.8734
Nursery	98.67%	0.9867	0.9867	0.9872

2) Tree Characteristics Analysis

Datasets	Tree Depth	Number of nodes
Mushroom	4	29
Tictactoe	7	281
Nursery	7	852

Datasets	Most Important Features
Mushroom	Odor , spore-print-color , habitat , gill-size , cap-color
Tictactoe	Top-middle-square , top-right-square , top-left-square , middle-right-square , middle-left-square , bottom-right-square , bottom-left-square , bottom-middle-square
Nursery	Social , housing , finance , form , children

Mushroom: Fewer splits , Small Tree

tictactoe: Hard separation , deep-medium tree.

Nursery: Many samples and more variation , very large tree.

3) Dataset-Specific Insights
Dataset 1 has strong, dominant features that allow the decision tree to classify all samples with a lower depth.

Dataset 2 requires deeper splits even with fewer samples, indicating more feature overlap or noise.

Dataset 3 while large, shows disparity between weighted and macro F1, suggesting that minority classes are harder to classify despite overall high accuracy.

4)
a) Algorithm Performance

i) Dataset 1 achieved 100% accuracy.

Reason: Small number of features and clear separability between classes allowed the decision tree to classify every sample correctly with very few splits.

ii)

Smaller datasets can lead to underfitting if the tree is shallow or overfitting if it's deep relative to the data.

Larger datasets allow better generalization, but may require deeper trees to capture complex patterns.

iii)

More features increase tree complexity because the algorithm has more potential splits.

If irrelevant features exist, the tree may overfit; highly informative features lead to better splits and higher accuracy.

b) Data Characteristics Impact

i)

Trees favor majority classes that is , high weighted accuracy but lower macro F1.

Minority classes might be misclassified, requiring deeper splits to handle rare cases.

ii)

Binary features: simpler splits, faster tree construction, less risk of overfitting.

Multi-valued features: more splits per node or , deeper tree, can capture more complex patterns but higher overfitting risk.

c) Practical Applications

i. Dataset 1

Use case: Medical diagnosis with clear symptom-to-disease mapping.

Interpretability: Easy to explain decisions to doctors or non-technical users.

ii. Dataset 2

Use case: Customer segmentation for marketing campaigns.

Interpretability: Medium complexity, can identify key decision paths for different customer types.

iii. Dataset 3

Use case: Fraud detection, or large-scale product classification.

Interpretability: Harder to explain every decision, but weighted accuracy makes sure that the majority patterns are captured; visualisations of common paths can help.