ML LAB-3

Name: DEVRAJ NAIK

SRN: PES2UG23CS167

CLASS: C

1)mushrooms.csv

```
PS C:\Users\Devraj\.vscode\PYTHON\ML_LAB\Lab_3> python test.py --ID EC_C_PES2UG23CS167_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-ab
ove-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', '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-a
bove-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spore-print-color', 'populatio
n', 'habitat']
Target: class
Framework: PYTORCH
Data type: <class 'torch.Tensor'>

bove-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spore-print-color', 'populatio
n', 'habitat']
Target: class
Framework: PYTORCH
Data type: <class 'torch.Tensor'>

============================================================
DECISION TREE CONSTRUCTION DEMO
============================================================
Total samples: 8124
bove-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spore-print-color', 'populatio
n', 'habitat']
Target: class
Framework: PYTORCH
```

```
============================================================
Total samples: 8124
Training samples: 6499
Total samples: 8124
Training samples: 6499
Testing samples: 1625
Training samples: 6499
Testing samples: 1625
Testing samples: 1625

Constructing decision tree using training data...

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
```

2) Nursery.csv

```
PS C:\Users\Devraj\.vscode\PYTHON\ML_LAB\Lab_3> python test.py --ID EC_C_PES2UG23CS167_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
```

3) tictactoe.csv

```
PS C:\Users\Devraj\.vscode\PYTHON\ML_LAB\Lab_3> python test.py --ID EC_C_PES2UG23CS167_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-sq
uare', '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-s
quare', '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
```

Description and Conclusions:

Dataset	Accuracy	Precision (W)	Recall (W)	F1 (W)	Precision (M)	Recall (M)	F1 (M)
Mushroom	**100.00%**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Nursery	98.67%	0.9876	0.9867	0.9872	0.7604	0.7654	0.7628
TicTacToe	87.30%	0.8741	0.8730	0.8734	0.8590	0.8638	0.8613

Dataset	Depth	Total Nodes	Leaf Nodes	Internal Nodes
Mushroom	4	29	24	5
Nursery	7	952	680	272
TicTacToe	7	281	180	101

- Mushroom dataset has shallow and small tree, but achieved strong feature discrimination. Here ‘odor’ is the most important feature split. No over/ underfitting as all test cases are passed with 100% accuracy (perfect study). No improvement is needed as everything is perfect here
- Nursery dataset’s tree is huge one with 952 nodes hence complex. ‘Has Nurse’ and ‘Finance’ is important one. Class distribution is little imbalanced, hence causing overfitting. Here class imbalance happens, so resampling, pruning can help
- Tictactoe’s tree has more depth but less (281 nodes) not that complex than Nursery but more than Mushrooms. Middle square has most effect on the outcome, generally. As tree is large, a little bit of overfitting, but still good compared to ‘Nursery’. For this Algorithm change can be good. Using random forest can help significantly. Otheriwise use ensemble methods