UE23CS352A-MACHINE LEARNING
LAB-3

NAME	SRN	SECTION
C PANSHUL REDDY	PES2UG23CS154	C

Output 1(mushrooms.csv):

```
PS C:\Users\cpans\OneDrive\Documents\code\pytorch_implementation> python test.py --ID EC_C_PES2UG23CS154_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', 'st
alk-surface-above-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spo
re-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', 's
talk-surface-above-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'sp
ore-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
```

The decision tree has been trained on the mushroom.csv dataset.It is an example of binary classification between poisonous and edible mushrooms.This model achieves a perfect accuracy of 100%.This indicates that the model perfectly classifies the data.

It also gives that the depth of the tree is just 4 and there are 29 nodes with 24 leaf nodes and 5 internal nodes indicating that the tree complexity is very low,the model is simple and efficient,low depth and low internal nodes signifies lesser number of key features are required.

Output 2(Nursery.csv):

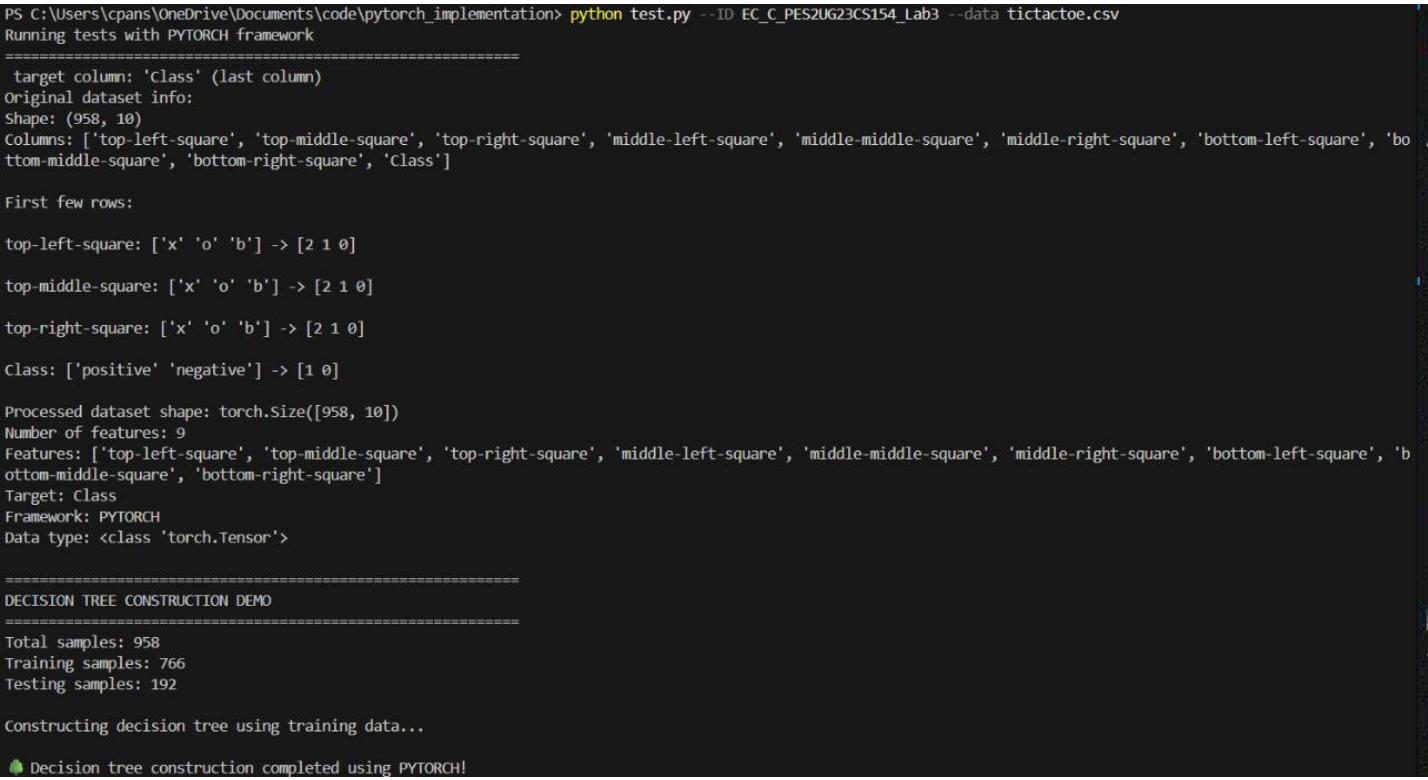
```
PS C:\Users\cpans\OneDrive\Documents\code\pytorch_implementation> python test.py --ID EC_C_PES2UG23CS154_Lab3 --data Nursery.csv
Running tests with PYTORCH framework
============================================================
 target column: 'class' (last column)
Original dataset info:
Shape: (12960, 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!
```

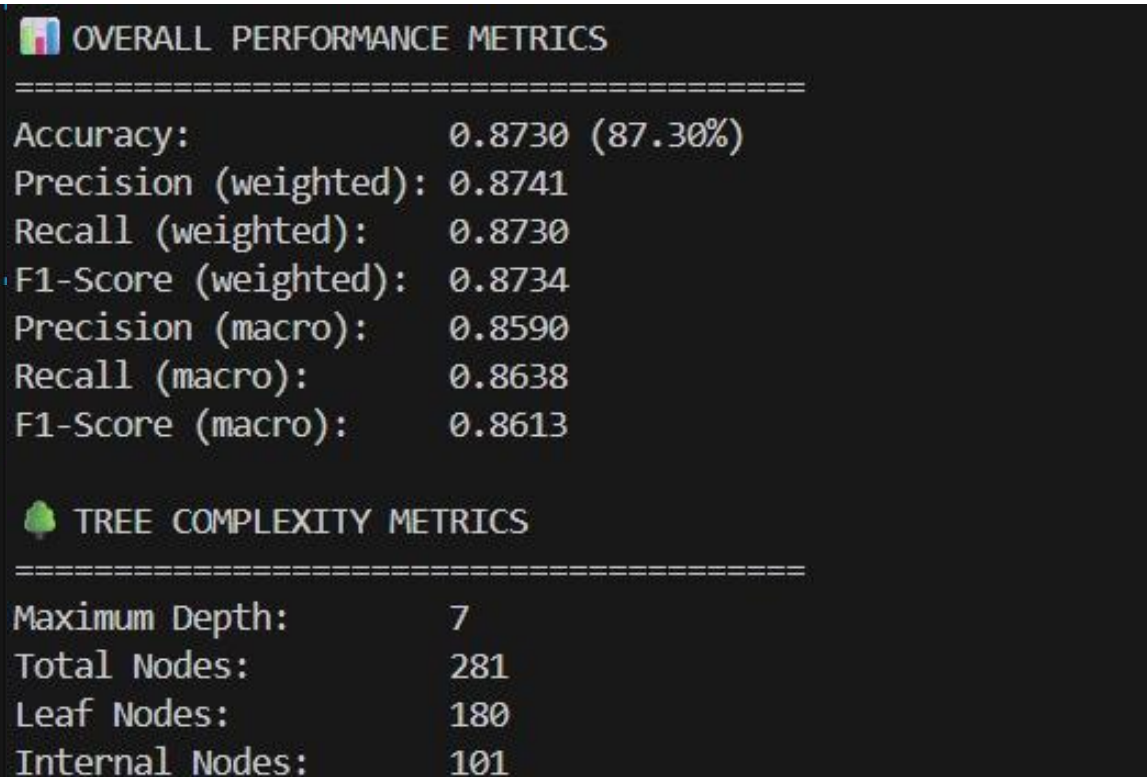
```
📊 OVERALL PERFORMANCE METRICS
==========================================
Accuracy:              0.9867 (98.67%)
Precision (weighted): 0.9876
Recall (weighted):    0.9867
F1-Score (weighted):  0.9872
Precision (macro):    0.7604
Recall (macro):       0.7654
F1-Score (macro):     0.7628

🌳 TREE COMPLEXITY METRICS
==========================================
Maximum Depth:         7
Total Nodes:           952
Leaf Nodes:            680
Internal Nodes:        272
```

This decision tree has been trained on the nursery.csv data set,this involves predicting a recommendation level for nursery school applications. This is a multiclass classification problem. The accuracy of this model is very high as well with 98.67% and a weighted f1 score of .9872.The hifh macro score signifies it does well on all classes even if there is imabalance.

This tree has a maximum depth of 7 with 952 total nodes 680 leaf and 272 internal nodes signifying a very complex tree.The high number of internal nodes indicates high number of decision points. The high accuracy and complex tree structure point to a strong fit with the training data.

Output 3(tictactoe.csv):





This decision tree has been trained on the tictactoe.csv data set, this is again a binary classification problem where the model predicts the outcome of the game. The models performance is respectable but not as good as those compared above because it has an accuracy fo just 87.3% with lower weighted metrics and macro scores.The low macro score indicates disparities across different class labels.

The tree has a max depth of 7 with 281 total nodes 180 of them being leaf and 101 internal. The complexity is substantial but the metrics show it is hard to solve it with a simple decision tree. Possibly due to higher degree of ambiguity or overlap between feature value and target class.