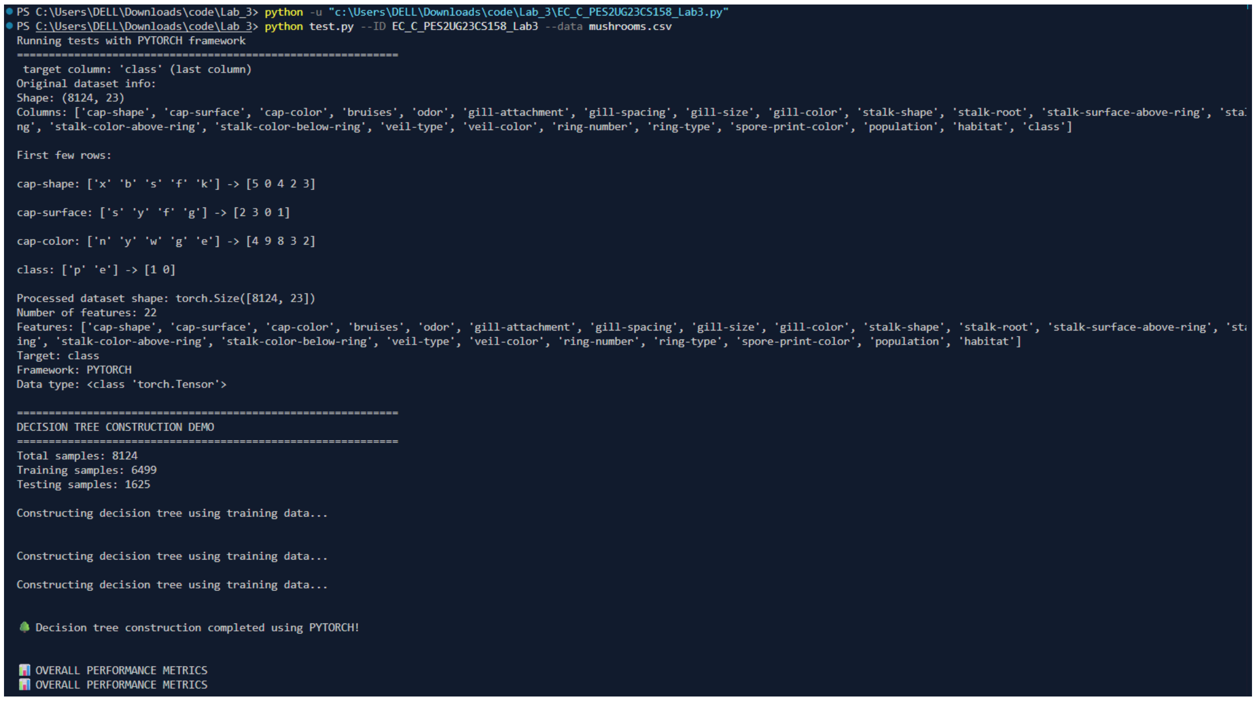
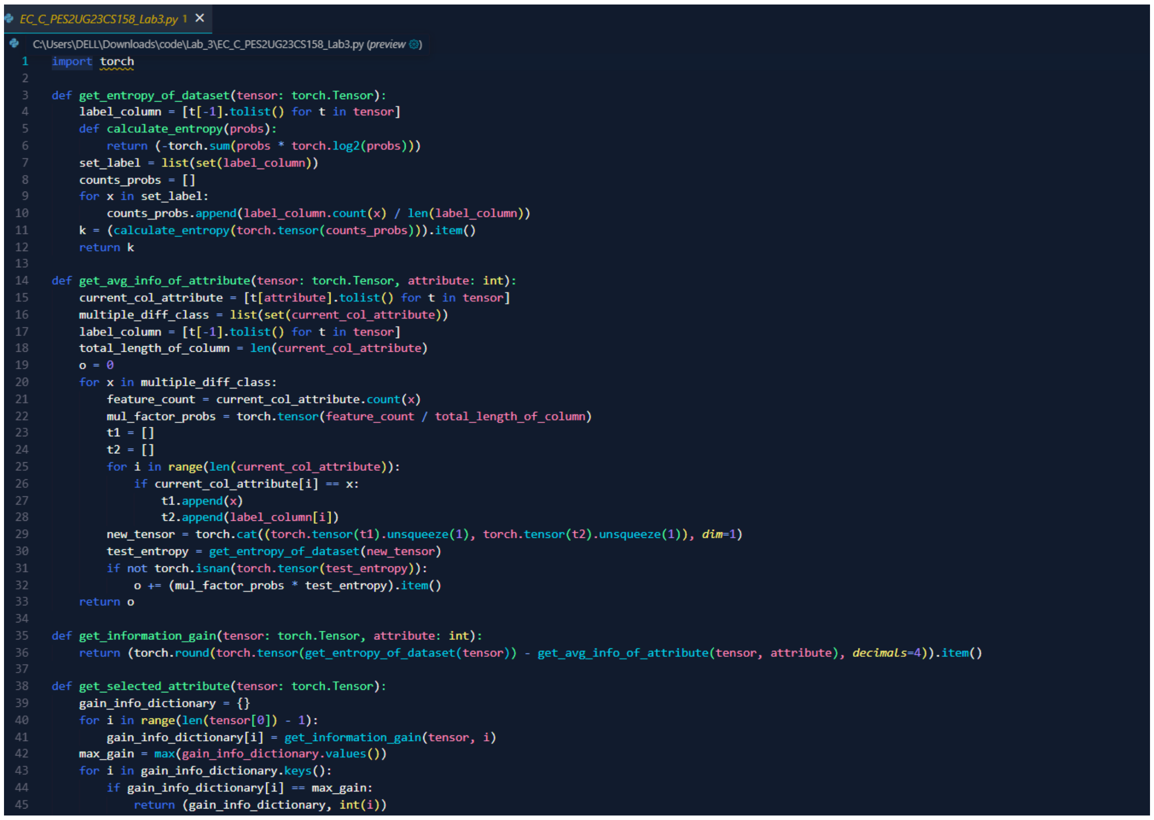


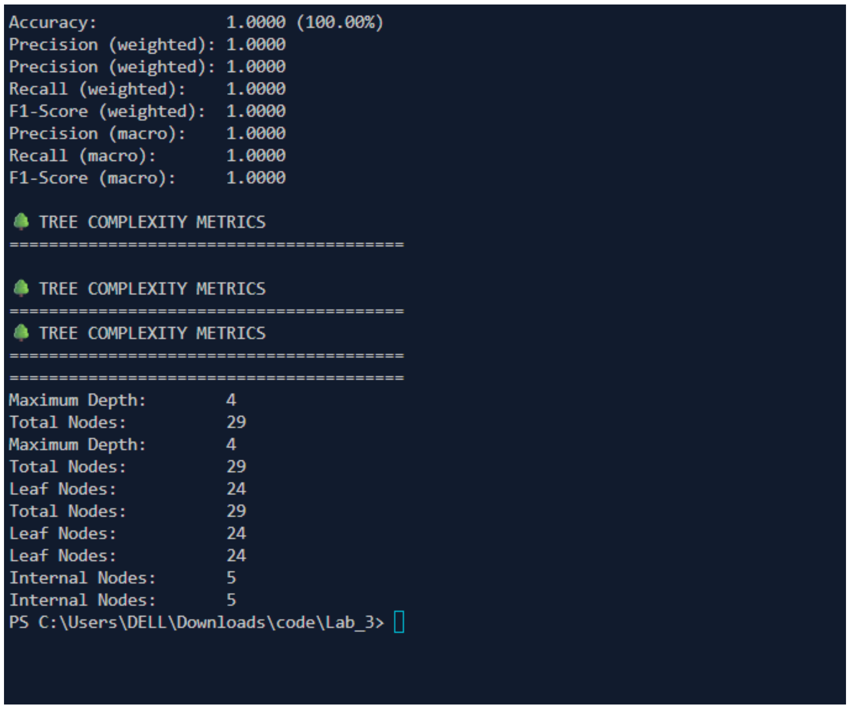
```
PS C:\Users\DELL\Downloads\code\Lab_3> python -u "c:\Users\DELL\Downloads\code\Lab_3\EC_C_PES2UG23CS158_Lab3.py"
PS C:\Users\DELL\Downloads\code\Lab_3> python test.py --ID EC_C_PES2UG23CS158_Lab3 --data mushrooms.csv
Running tests with PYTORCH framework
============================================================
 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', 'sta
ng', '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', 'sta
ing', '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...


Constructing decision tree using training data...

Constructing decision tree using training data...


🌲 Decision tree construction completed using PYTORCH!


📊 OVERALL PERFORMANCE METRICS
📊 OVERALL PERFORMANCE METRICS
```



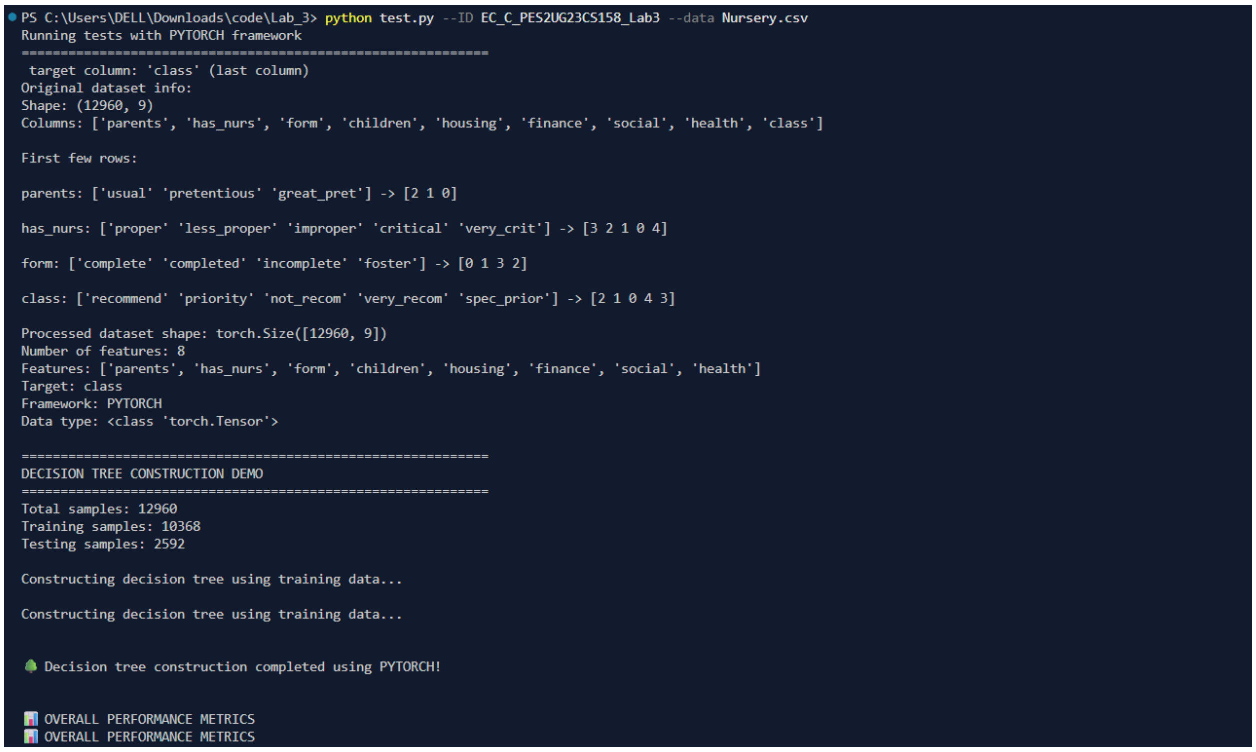
EC_C_PES2UG23CS158_Lab3.py 1 ✕

C:\Users\DELL\Downloads\code\Lab_3\EC_C_PES2UG23CS158_Lab3.py (preview)

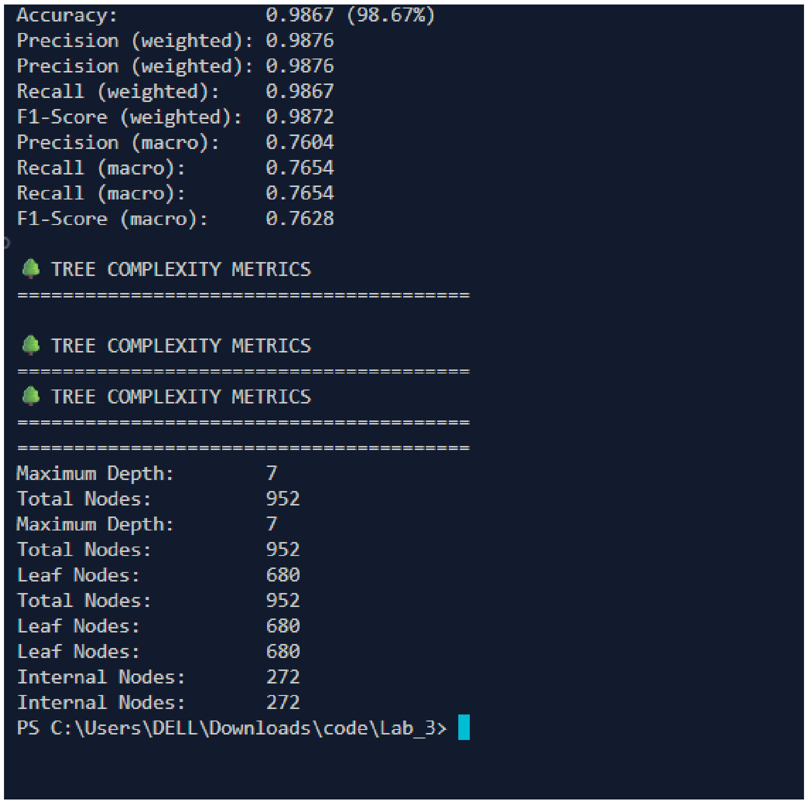
```python
import torch

def get_entropy_of_dataset(tensor: torch.Tensor):
    label_column = [t[-1].tolist() for t in tensor]
    def calculate_entropy(probs):
        return (-torch.sum(probs * torch.log2(probs)))
    set_label = list(set(label_column))
    counts_probs = []
    for x in set_label:
        counts_probs.append(label_column.count(x) / len(label_column))
    k = (calculate_entropy(torch.tensor(counts_probs))).item()
    return k

def get_avg_info_of_attribute(tensor: torch.Tensor, attribute: int):
    current_col_attribute = [t[attribute].tolist() for t in tensor]
    multiple_diff_class = list(set(current_col_attribute))
    label_column = [t[-1].tolist() for t in tensor]
    total_length_of_column = len(current_col_attribute)
    o = 0
    for x in multiple_diff_class:
        feature_count = current_col_attribute.count(x)
        mul_factor_probs = torch.tensor(feature_count / total_length_of_column)
        t1 = []
        t2 = []
        for i in range(len(current_col_attribute)):
            if current_col_attribute[i] == x:
                t1.append(x)
                t2.append(label_column[i])
        new_tensor = torch.cat((torch.tensor(t1).unsqueeze(1), torch.tensor(t2).unsqueeze(1)), dim=1)
        test_entropy = get_entropy_of_dataset(new_tensor)
        if not torch.isnan(torch.tensor(test_entropy)):
            o += (mul_factor_probs * test_entropy).item()
    return o

def get_information_gain(tensor: torch.Tensor, attribute: int):
    return (torch.round(torch.tensor(get_entropy_of_dataset(tensor) - get_avg_info_of_attribute(tensor, attribute), decimals=4)).item()

def get_selected_attribute(tensor: torch.Tensor):
    gain_info_dictionary = {}
    for i in range(len(tensor[0]) - 1):
        gain_info_dictionary[i] = get_information_gain(tensor, i)
    max_gain = max(gain_info_dictionary.values())
    for i in gain_info_dictionary.keys():
        if gain_info_dictionary[i] == max_gain:
            return (gain_info_dictionary, int(i))
```



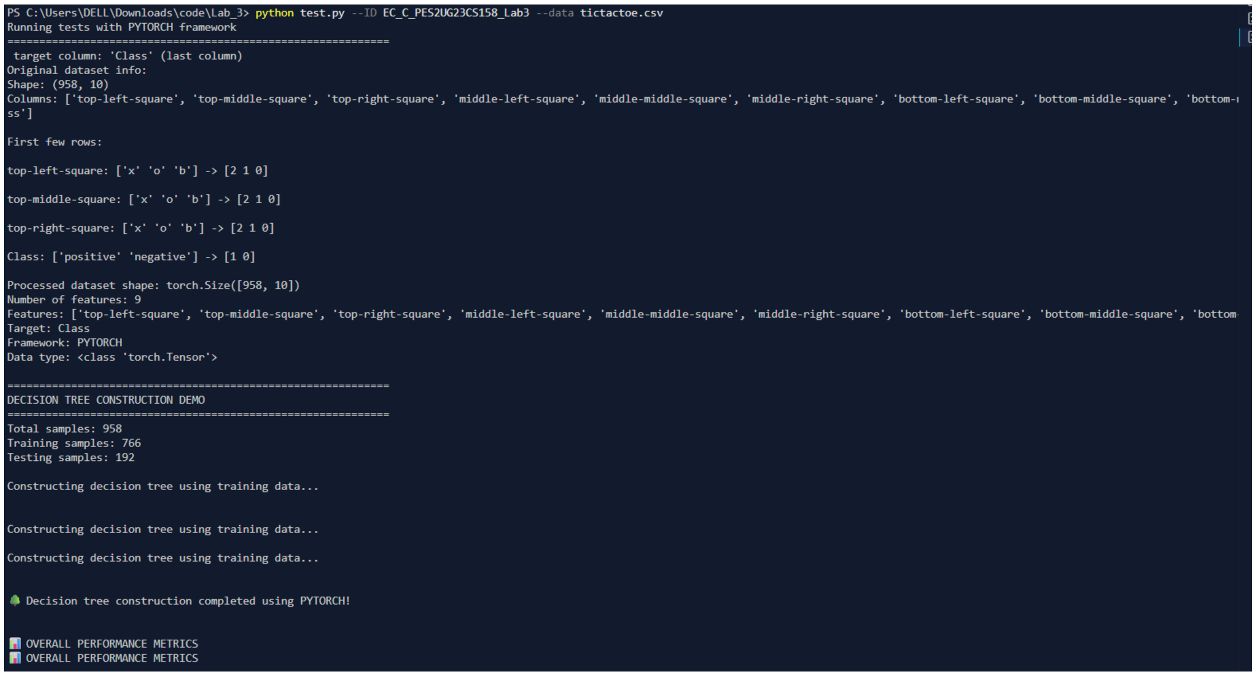
```
Accuracy:             1.0000 (100.00%)
Precision (weighted): 1.0000
Precision (weighted): 1.0000
Recall (weighted):    1.0000
F1-Score (weighted):  1.0000
Precision (macro):    1.0000
Recall (macro):       1.0000
F1-Score (macro):     1.0000

🌳 TREE COMPLEXITY METRICS
========================================

🌳 TREE COMPLEXITY METRICS
========================================
🌳 TREE COMPLEXITY METRICS
========================================
========================================
Maximum Depth:        4
Total Nodes:          29
Maximum Depth:        4
Total Nodes:          29
Leaf Nodes:           24
Total Nodes:          29
Leaf Nodes:           24
Leaf Nodes:           24
Internal Nodes:       5
Internal Nodes:       5
PS C:\Users\DELL\Downloads\code\Lab_3>
```



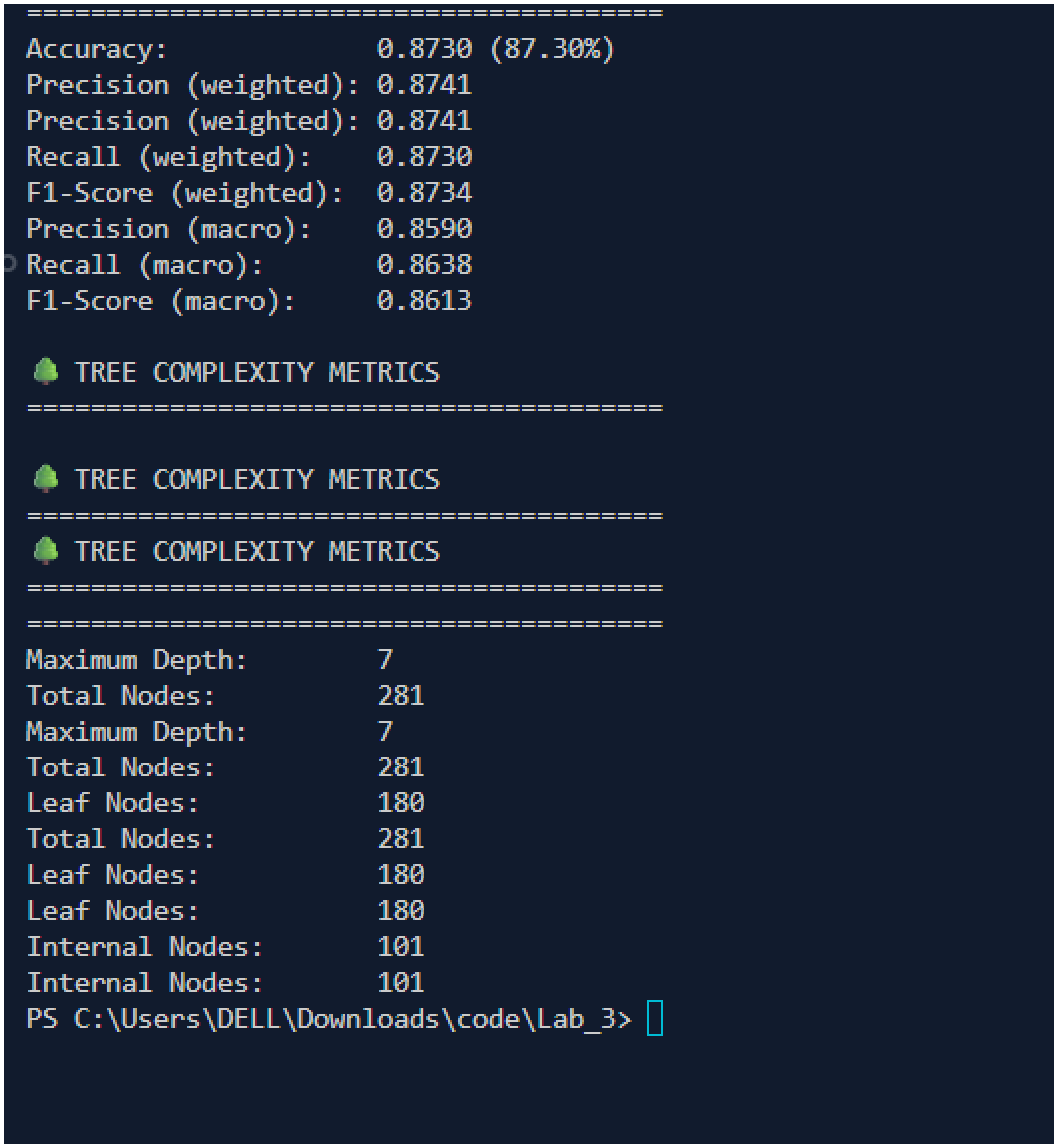
```
● PS C:\Users\DELL\Downloads\code\Lab_3> python test.py --ID EC_C_PES2UG23CS158_Lab3 --data Nursery.csv
  Running tests with PYTORCH framework
  ============================================================
   target column: 'class' (last column)
  Original dataset info:
  Shape: (12960, 9)
  Columns: ['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...

  Constructing decision tree using training data...


   🌳 Decision tree construction completed using PYTORCH!


   📊 OVERALL PERFORMANCE METRICS
   📊 OVERALL PERFORMANCE METRICS
```



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

🌳 TREE COMPLEXITY METRICS
========================================

🌳 TREE COMPLEXITY METRICS
========================================
🌳 TREE COMPLEXITY METRICS
========================================
========================================
Maximum Depth:        7
Total Nodes:          952
Maximum Depth:        7
Total Nodes:          952
Leaf Nodes:           680
Total Nodes:          952
Leaf Nodes:           680
Leaf Nodes:           680
Internal Nodes:       272
Internal Nodes:       272
PS C:\Users\DELL\Downloads\code\Lab_3>
```



```
PS C:\Users\DELL\Downloads\code\Lab_3> python test.py --ID EC_C_PES2UG23CS158_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-
ss']

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
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...


Constructing decision tree using training data...

Constructing decision tree using training data...


🌳 Decision tree construction completed using PYTORCH!


📊 OVERALL PERFORMANCE METRICS
📊 OVERALL PERFORMANCE METRICS
```



```
========================================
Accuracy:              0.8730 (87.30%)
Precision (weighted): 0.8741
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
========================================

🌳 TREE COMPLEXITY METRICS
========================================
🌳 TREE COMPLEXITY METRICS
========================================
========================================
Maximum Depth:         7
Total Nodes:           281
Maximum Depth:         7
Total Nodes:           281
Leaf Nodes:            180
Total Nodes:           281
Leaf Nodes:            180
Leaf Nodes:            180
Internal Nodes:        101
Internal Nodes:        101
PS C:\Users\DELL\Downloads\code\Lab_3>
```