

**UE23CS352A-MACHINE LEARNING**

**LAB-3**

**NAME : C Yogesh Reddy**

**SRN : PES2UG23CS159**

**SEC : C**

## Output1(mushrooms.csv):

```
[Notice] To update your Python environment to use pip, first run: python -m pip install --upgrade pip
PS C:\Users\yoges\Downloads\all (2)\all\code\pytorch_implementation> python test.py --ID EC_C_PES2UG23CS159_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-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-above-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']
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
```

```
=====
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\yoges\Downloads\all (2)\all\code\pytorch_implementation>
```

A decision tree model trained on the mushroom dataset can perfectly classify whether a mushroom is edible or poisonous using a very simple structure. It needs only a few important features and, with just four levels and 29 branches, it gets every prediction right. This means you can achieve flawless

**results with a small and straightforward decision process.**

## **Output2(Nursery.csv):**

```
PS C:\Users\yoges\Downloads\all (2)\all\code\pytorch_implementation> python test.py --ID EC_C_PES2UG23CS159_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\yoges\Downloads\all (2)\all\code\pytorch_implementation>
```

The decision tree trained on the nursery.csv dataset predicts recommendation levels for nursery admissions, which makes it a multiclass classification task. The model achieves a very high accuracy of 98.67% and a weighted F1 score of 0.9872, showing it's highly effective overall. The high macro F1 score means the model performs well for every class, even if the classes aren't balanced. This decision tree is quite complicated, reaching a depth of 7 and containing 952 nodes in total—272 points where decisions are made and 680 final outcomes (leaves). The large number of internal nodes reflects many decision steps. Such high accuracy along with this level of complexity suggests that the model fits the training data very closely.

### Output3(tictactoe.csv):

```
PS C:\Users\yoges\Downloads\all (2)\all\code\pytorch_implementation> python test.py --ID EC_C_PES2UG23CS159_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...

tree Decision tree construction completed using PYTORCH!

chart 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 TREE COMPLEXITY METRICS
=====
Maximum Depth: 7
Total Nodes: 281
Leaf Nodes: 180
Internal Nodes: 101
PS C:\Users\yoges\Downloads\all (2)\all\code\pytorch_implementation>
```

A decision tree was built to predict the result of a tic-tac-toe game, which involves choosing between two possible outcomes. The model does a decent job, with an accuracy of 87.3%, but its performance isn't as strong as the other examples. Its lower macro score means it isn't equally good at predicting both outcomes. The tree is fairly complex, with a depth of 7 and a total of 281 branches, but the results suggest that the problem is tougher—likely because the features don't always clearly separate the outcomes, making it difficult for the model to consistently get the prediction right.

**END**