

Machine Learning Lab Assignment - 1

Name: Dhruv Maheshwari

Sem 5 – C

Date: 19th Aug 2025

Link to Kaggle Notebook:

<https://www.kaggle.com/code/dhruvmaheshwari2004/pes2ug23cs173>

```
# For Mushroom dataset
!python /kaggle/input/machinelearninglab/test.py --ID EC_C_PES2UG23CS173_Lab3 --data /kaggle/input/machinelearninglab/mushrooms.csv

# For Tic-Tac-Toe dataset
!python /kaggle/input/machinelearninglab/test.py --ID EC_C_PES2UG23CS173_Lab3 --data /kaggle/input/machinelearninglab/tictactoe.csv

# For Nursery dataset
!python /kaggle/input/machinelearninglab/test.py --ID EC_C_PES2UG23CS173_Lab3 --data /kaggle/input/machinelearninglab/Nursery.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: ['s' '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
=====
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
Running tests with PYTORCH framework
=====
target column: 'class' (last column)
Original dataset info:
Shape: (958, 18)
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']
```

Entropy Calculation: To determine a dataset's entropy, a function is developed. The degree of class mixing in a given set of examples can be ascertained by measuring the impurity or randomness of the data using entropy.

Average Information: The average information of a particular attribute is determined by another function. In essence, this is a weighted average of the entropy of each subset that was produced by dividing the dataset according to that characteristic.

Information Gain: A third function determines the information gain for each attribute using the two functions mentioned above. The decrease in entropy that arises from dividing the data according to a specific feature is measured by information gain. For a split, the best attribute to use is the one with the highest information gain.

Attribute Selection: The last function iterates through all of the features that are available and chooses the one with the greatest information gain in order to determine which attribute should be used for the subsequent split in the tree.

```
PES2UG23CS173 Draft saved
File Edit View Run Settings Add-ons Help
+ - [Icons] Run All Code
● Draft Session off (run a cell to start) [Icons]

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 (87.10%)
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

PES2UG23CS173 Draft saved
File Edit View Run Settings Add-ons Help
+ - [Icons] Run All Code
● Draft Session off (run a cell to start) [Icons]

● TREE COMPLEXITY METRICS
=====
Maximum Depth: 7
Total Nodes: 281
Leaf Nodes: 180
Internal Nodes: 101
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): 0.7684
Recall (macro): 0.7654
F1-Score (macro): 0.7628

● TREE COMPLEXITY METRICS
=====
Maximum Depth: 7
Total Nodes: 952
Leaf Nodes: 680
Internal Nodes: 272
+ Code + Markdown

[7]: # To visualize the tree for the Mushroom dataset
!python /kaggle/input/machinelearninglab/test.py --ID EC_C_PES2UG23CS173_Lab3 --data /kaggle/input/machinelearninglab/mushrooms.csv --print-tree

Running tests with PYTORCH framework
=====
target column: 'class' (last column)
Original dataset info:
Shape: (4124, 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']
```

Execution: The implemented code is run on each dataset using commands to record performance metrics like F1-score, recall, accuracy, and precision.

Analysis: A thorough analysis is based on the results of these tests. A thorough comparison of the findings from all three datasets is included in the notebook. This analysis includes:

Performance Metrics: A comparison of each dataset's classification metrics.

```
PES2UG23CS173 Draft saved
File Edit View Run Settings Add-ons Help

+ - X Copy Paste Run All Code

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 0 3 2]
class: ['g' '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!

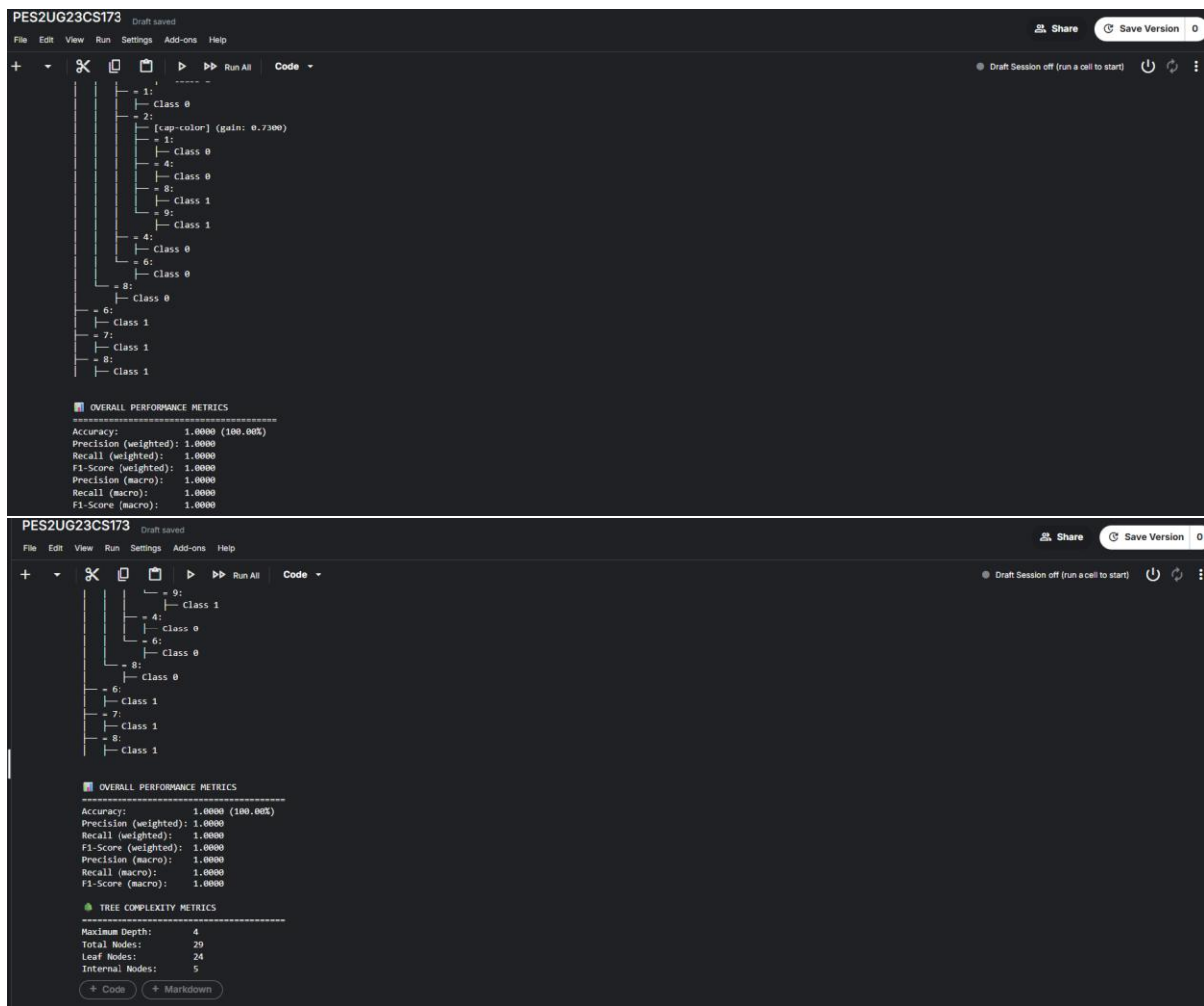
PES2UG23CS173 Draft saved
File Edit View Run Settings Add-ons Help

+ - X Copy Paste Run All Code

DECISION TREE STRUCTURE
=====
Root [odor] (gain: 0.9083)
- 0:
  - Class 0
- 1:
  - Class 1
- 2:
  - Class 1
- 3:
  - Class 0
- 4:
  - Class 1
- 5:
  - [spore-print-color] (gain: 0.1469)
    - 0:
      - Class 0
    - 1:
      - Class 0
    - 2:
      - Class 0
    - 3:
      - Class 0
    - 4:
      - Class 0
    - 5:
      - Class 1
    - 6:
      - Class 1
  - [habitat] (gain: 0.2217)
    - 0:
      - [gill-size] (gain: 0.7642)
        - 0:
          - Class 0
        - 1:
          - Class 1
    - 1:
      - Class 0
    - 2:
      - Class 1
    - 3:
      - Class 0
    - 4:
      - Class 1
    - 5:
      - Class 0
    - 6:
      - Class 1
    - 7:
      - Class 0
    - 8:
      - Class 1
    - 9:
      - Class 0
    - 10:
      - Class 1
    - 11:
      - Class 0
    - 12:
      - Class 1
    - 13:
      - Class 0
    - 14:
      - Class 1
    - 15:
      - Class 0
    - 16:
      - Class 1
    - 17:
      - Class 0
    - 18:
      - Class 1
    - 19:
      - Class 0
    - 20:
      - Class 1
    - 21:
      - Class 0
    - 22:
      - Class 1
```

Tree Characteristics: An analysis of the depth and total number of nodes in the resultant tree.

Feature Importance: An examination of the features (the tree's root and early splits) that were deemed most crucial for decision-making.



Insights: An end-of-term report explaining how the performance and shape of the decision tree varied in response to dimensions such as dataset size, feature dimension, and class distribution for each particular problem. The report also indicates the benefits of applying an interpretable model such as a decision tree.