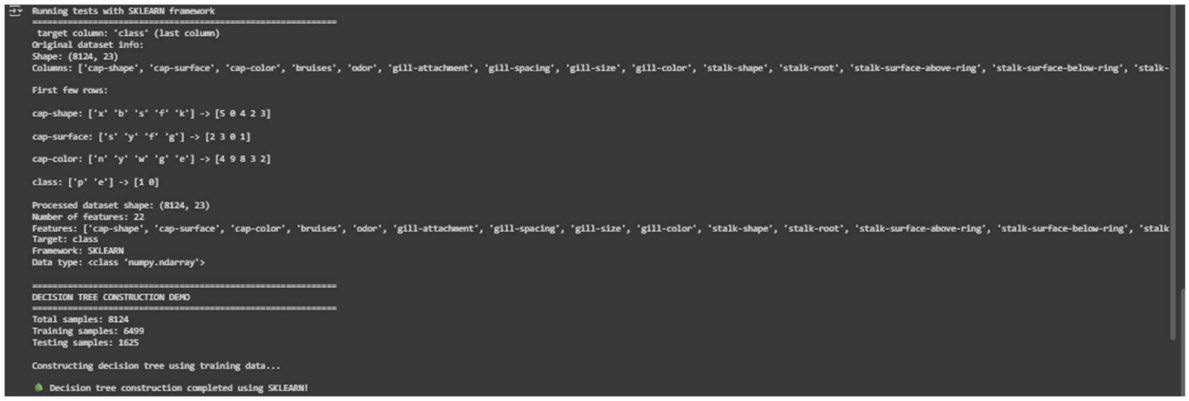
ML Lab

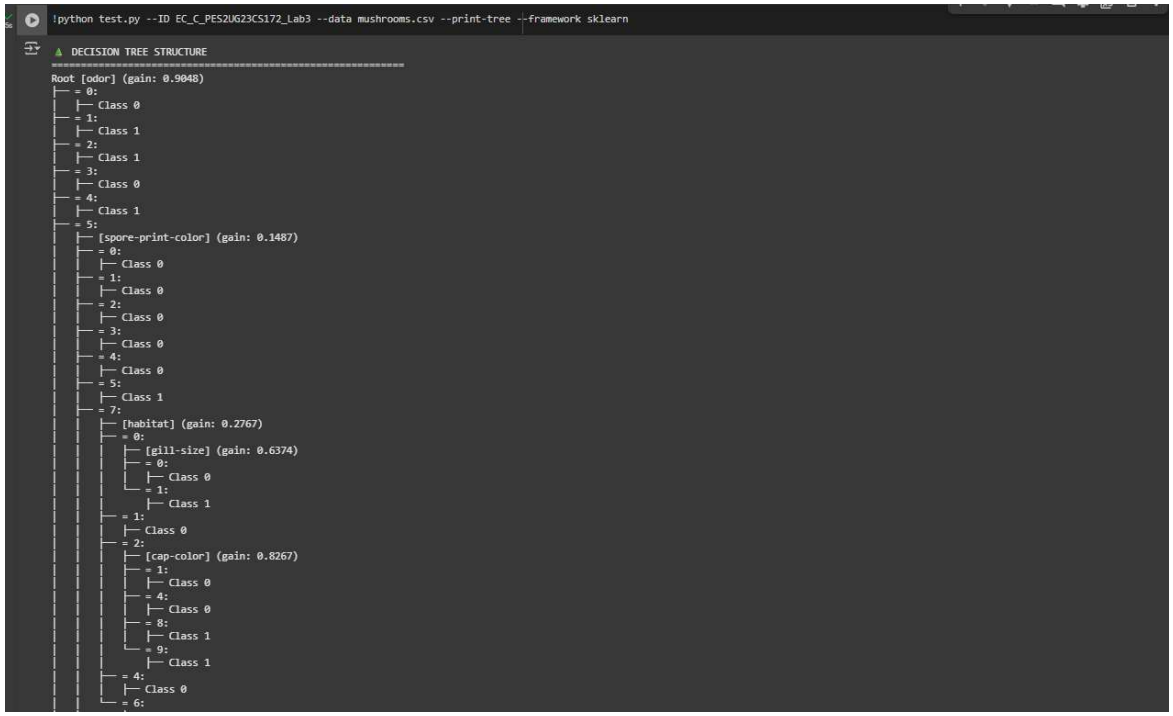
Week-3

Name: Dhruv Hegde	SRN: PES2UG23CS172	Section: 5C

Week: 3

1.Mushrooms



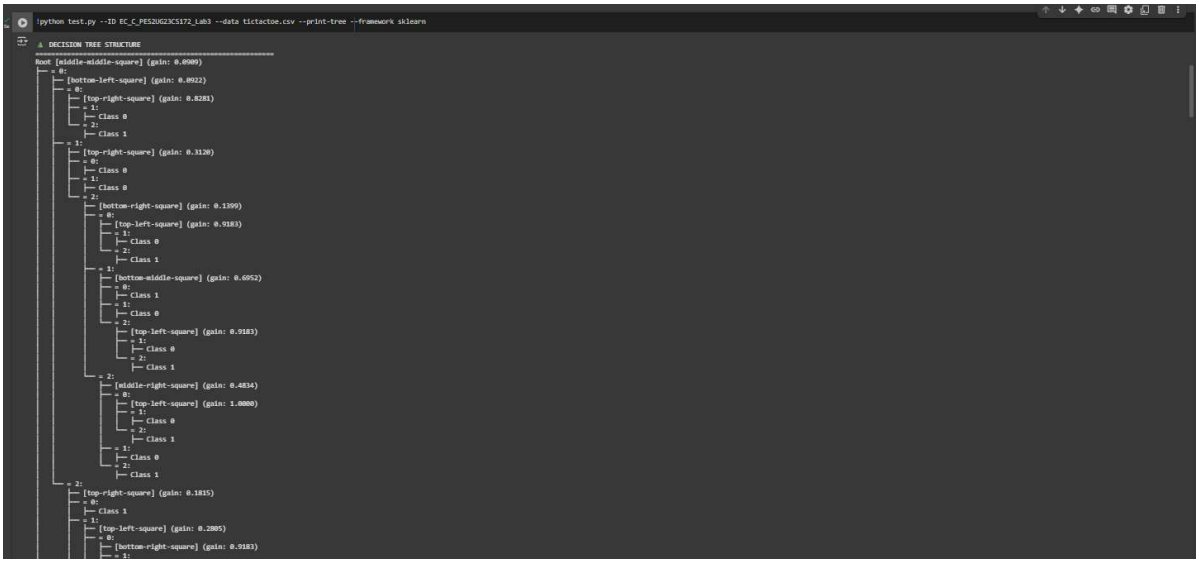


...

```
    |   |   ├── = 4:
    |   |   |   ├── Class 0
    |   |   └── = 6:
    |   |       ├── Class 0
    |   └── = 8:
    |       ├── Class 0
    ├── = 6:
    |   ├── Class 1
    ├── = 7:
    |   ├── Class 1
    ├── = 8:
    |   ├── Class 1


    📊 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.Tic-tac-toe

```
Running tests with SKLEARN 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: (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: SKLEARN
Data type: <class 'numpy.ndarray'>

============================================================
DECISION TREE CONSTRUCTION DEMO
============================================================
Total samples: 958
Training samples: 766
Testing samples: 192

Constructing decision tree using training data...

🌳 Decision tree construction completed using SKLEARN!
```


!python test.py --ID EC_C_PES2UG23CS172_Lab3 --data tictactoe.csv --print-tree --framework sklearn
DECISION TREE STRUCTURE
Root [middle-middle-square] (gain: 0.0909)
= 0:
[bottom-left-square] (gain: 0.0922)
= 0:
[top-right-square] (gain: 0.8281)
= 1:
Class 0
= 2:
Class 1
= 1:
[top-right-square] (gain: 0.3128)
= 0:
Class 0
= 1:
Class 0
= 2:
[bottom-right-square] (gain: 0.1399)
= 0:
[top-left-square] (gain: 0.9183)
= 1:
Class 0
= 2:
Class 1
= 1:
[bottom-middle-square] (gain: 0.6952)
= 0:
Class 1
= 1:
Class 0
= 2:
[top-left-square] (gain: 0.9183)
= 1:
Class 0
= 2:
Class 1
= 2:
[middle-right-square] (gain: 0.4834)
= 0:
[top-left-square] (gain: 1.0000)
= 1:
Class 0
= 2:
Class 1
= 1:
Class 0
= 2:
Class 1
= 2:
[top-right-square] (gain: 0.1815)
= 0:
Class 1
= 1:
[top-left-square] (gain: 0.2805)
= 0:
[bottom-right-square] (gain: 0.9183)
= 1:

•••


Class 1
= 1:
[middle-right-square] (gain: 0.5020)
= 0:
[top-left-square] (gain: 0.3219)
= 0:
Class 1
= 1:
[top-middle-square] (gain: 1.0000)
= 1:
Class 0
= 2:
Class 1
= 2:
Class 1
= 1:
Class 0
= 2:
[middle-left-square] (gain: 0.5850)
= 0:
Class 0
= 1:
[top-left-square] (gain: 1.0000)
= 1:
Class 1
= 2:
Class 0
= 2:
Class 1
= 2:
[middle-right-square] (gain: 0.6962)
= 0:
Class 1
= 1:
Class 1
= 2:
Class 0
= 2:
Class 1
OVERALL PERFORMANCE METRICS
Accuracy: 0.8836 (88.36%)
Precision (weighted): 0.8827
Recall (weighted): 0.8836
F1-Score (weighted): 0.8822
Precision (macro): 0.8784
Recall (macro): 0.8680
F1-Score (macro): 0.8680
TREE COMPLEXITY METRICS
Maximum Depth: 7
Total Nodes: 260
Leaf Nodes: 165
Internal Nodes: 95

3.Nursery

```
Running tests with SKLEARN 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: (12960, 9)
Number of features: 8
Features: ['parents', 'has_nurs', 'form', 'children', 'housing', 'finance', 'social', 'health']
Target: class
Framework: SKLEARN
Data type: <class 'numpy.ndarray'>

============================================================
DECISION TREE CONSTRUCTION DEMO
============================================================
Total samples: 12960
Training samples: 10368
Testing samples: 2592

Constructing decision tree using training data...

 Decision tree construction completed using SKLEARN!
```

```
!python test.py --ID EC_C_PES2UG23CS172_Lab3 --data Nursery.csv --print-tree --framework sklearn

 DECISION TREE STRUCTURE
============================================================
Root [health] (gain: 0.9597)
├─ = 0:
│   ├─ Class 0
├─ = 1:
│   ├─ [has_nurs] (gain: 0.3648)
│   ├─ = 0:
│   │   ├─ [parents] (gain: 0.1629)
│   │   ├─ = 0:
│   │   │   ├─ [children] (gain: 0.0288)
│   │   │   ├─ = 0:
│   │   │   │   ├─ [form] (gain: 0.1164)
│   │   │   │   ├─ = 0:
│   │   │   │   │   ├─ [housing] (gain: 0.4828)
│   │   │   │   │   ├─ = 0:
│   │   │   │   │   │   ├─ [finance] (gain: 0.9710)
│   │   │   │   │   │   ├─ = 0:
│   │   │   │   │   │   │   ├─ Class 1
│   │   │   │   │   │   └─ = 1:
│   │   │   │   │   │       ├─ Class 3
│   │   │   │   │   ├─ = 1:
│   │   │   │   │   │   ├─ Class 3
│   │   │   │   │   └─ = 2:
│   │   │   │   │       ├─ Class 3
│   │   │   │   ├─ = 1:
│   │   │   │   │   ├─ Class 3
│   │   │   │   ├─ = 2:
│   │   │   │   │   ├─ Class 3
│   │   │   │   └─ = 3:
│   │   │   │       ├─ Class 3
│   │   │   ├─ = 1:
│   │   │   │   ├─ Class 3
│   │   │   ├─ = 2:
│   │   │   │   ├─ Class 3
│   │   │   └─ = 3:
│   │   │       ├─ Class 3
│   │   ├─ = 1:
│   │   │   ├─ [form] (gain: 0.0277)
│   │   │   ├─ = 0:
│   │   │   │   ├─ [children] (gain: 0.1152)
│   │   │   │   ├─ = 0:
│   │   │   │   │   ├─ [housing] (gain: 0.4828)
│   │   │   │   │   ├─ = 0:
│   │   │   │   │   │   ├─ [finance] (gain: 0.9710)
│   │   │   │   │   │   ├─ = 0:
│   │   │   │   │   │   │   ├─ Class 1
│   │   │   │   │   │   └─ = 1:
│   │   │   │   │   │       ├─ Class 3
│   │   │   │   │   ├─ = 1:
│   │   │   │   │   │   ├─ Class 3
│   │   │   │   │   └─ = 2:
│   │   │   │   │       ├─ Class 3
│   │   │   │   ├─ = 1:
│   │   │   │   │   ├─ Class 3
│   │   │   │   ├─ = 2:
│   │   │   │   │   ├─ Class 3
```

...

```
│   │   │   │   │   ├─ = 0:
│   │   │   │   │   │   ├─ Class 1
│   │   │   │   │   ├─ = 1:
│   │   │   │   │   │   ├─ Class 3
│   │   │   │   │   ├─ = 2:
│   │   │   │   │   │   ├─ Class 3
│   │   │   │   │   └─ = 3:
│   │   │   │   │       ├─ Class 3
│   │   │   │   ├─ = 1:
│   │   │   │   │   ├─ Class 3
│   │   │   │   ├─ = 2:
│   │   │   │   │   ├─ Class 3
│   │   │   │   └─ = 3:
│   │   │   │       ├─ Class 3
│   │   └─ = 2:
│   │       ├─ [children] (gain: 0.4463)
│   │       ├─ = 0:
│   │       │   ├─ [form] (gain: 0.8315)
│   │       │   ├─ = 0:
│   │       │   │   ├─ Class 1
│   │       │   ├─ = 1:
│   │       │   │   ├─ Class 1
│   │       │   ├─ = 2:
│   │       │   │   ├─ Class 3
│   │       │   └─ = 3:
│   │       │       ├─ Class 1
│   │       ├─ = 1:
│   │       │   ├─ [form] (gain: 1.0000)
│   │       │   ├─ = 0:
│   │       │   │   ├─ Class 1
│   │       │   ├─ = 1:
│   │       │   │   ├─ Class 1
│   │       │   ├─ = 2:
│   │       │   │   ├─ Class 3
│   │       │   └─ = 3:
│   │       │       ├─ Class 3
│   │       ├─ = 2:
│   │       │   ├─ Class 3
│   │       └─ = 3:
│   │           ├─ Class 3

 OVERALL PERFORMANCE METRICS
==================================================
Accuracy:           0.9887 (98.87%)
Precision (weighted): 0.9888
Recall (weighted):    0.9887
F1-Score (weighted):  0.9887
Precision (macro):    0.9577
Recall (macro):       0.9576
F1-Score (macro):     0.9576

 TREE COMPLEXITY METRICS
==================================================
Maximum Depth:        7
Total Nodes:          983
Leaf Nodes:           703
Internal Nodes:       280
```

1.Compare the following metrics across all three datasets:

Dataset	Accuracy	Precision		Recall		F1-Score	
		Weighted	Macro	Weighted	Macro	Weighted	Macro
Mushrooms	1.00	1.0	1.0	1.0	1.0	1.0	1.0
Tic-tac-toe	.8836	.8827	.8784	.8836	.8600	.8822	.8680
Nursery	.9887	.9888	.9577	.9887	.9576	.9887	.9576

2.Tree Characteristics Analysis

	Mushrooms	Tic-tac-toe	Nursery
• Tree Depth:	4	7	7
• Number of Nodes:	29	260	983
• Most Important Features:	22	9	8
•Complexity	low	high	medium

3. Dataset-Specific Insights

• Feature Importance:	Odor	Centre-square	Health
• Class Distribution:	Balanced	Unbalanced	Unbalanced
• Overfitting Indicators:	No	Yes(slight)	Yes

4. Comparative Analysis Report

a) Algorithm Performance:

a. • Which dataset achieved the highest accuracy and why?

Mushroom achieved the highest accuracy of 100.This is because one attribute clearly differentiates the two target classes.

b. • How does dataset size affect performance?

Size does play a major factor in performance. When the size is bigger there is greater number of test samples to go through which increases accuracy but also can increase the complexity and error margins.

c. • What role does the number of features play?

More the number of features more is the complexity for accurate results. Whereas few important feature's help's in more accurate results.

b) Data Characteristics Impact:

• How does class imbalance affect tree construction?

Class imbalance might lead to bias towards majority classes. Therefore, it's better to keep it balanced.

- **Which types of features (binary vs multi-valued) work better?**

  Binary works better as it gives simpler balanced trees compared to multi-valued.

c)Practical Applications:

- **For which real-world scenarios is each dataset type most relevant?**

  Tic-tac-toe-Game behavior modeling
  Mushroom-Food safety checks
  Nursery-Admission according to some factors

- **What are the interpretability advantages for each domain?**

  Decision trees provide clear “if-then” rules, making it easier to interpret factors for a certain dataset

- **How would you improve performance for each dataset?**

  Performance can be improved by the help of pruning for Mushrooms dataset. In the tic-tac-toe we need to handle imbalance. In Nursery pruning and ensemble methods can be used.