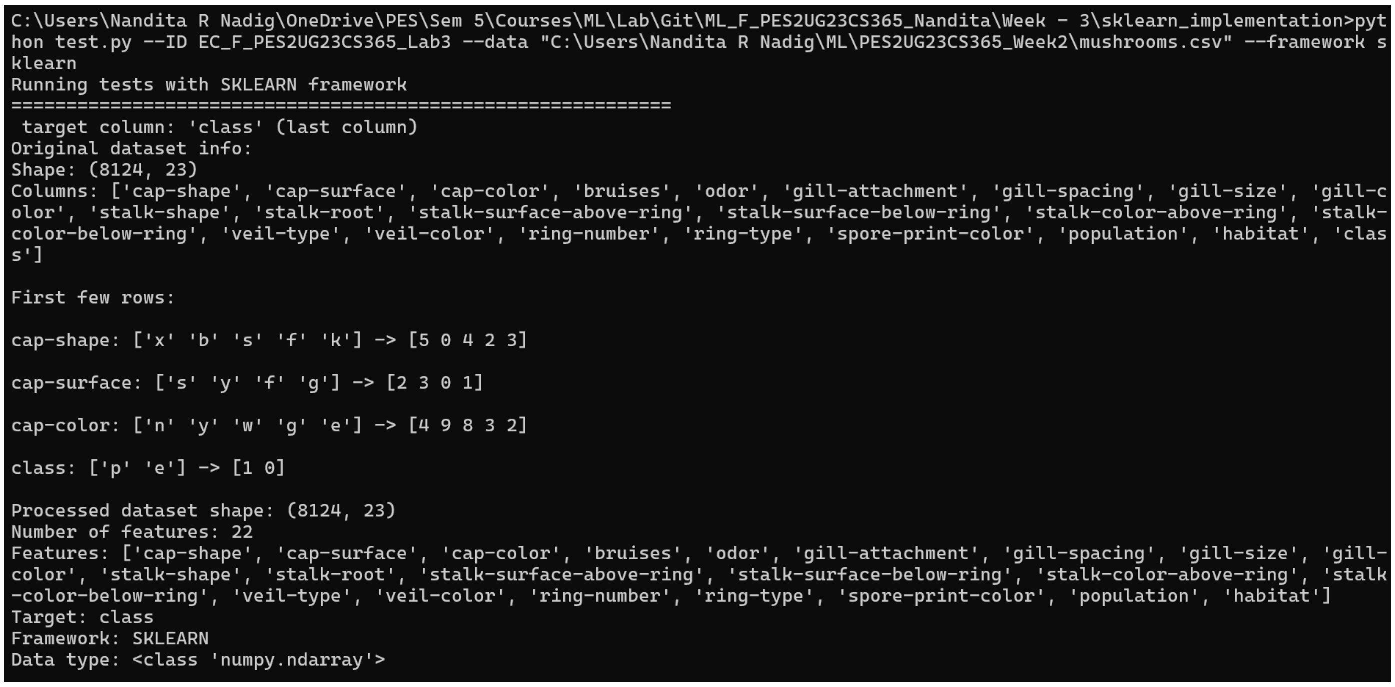
Machine Learning – Lab

Week – 3

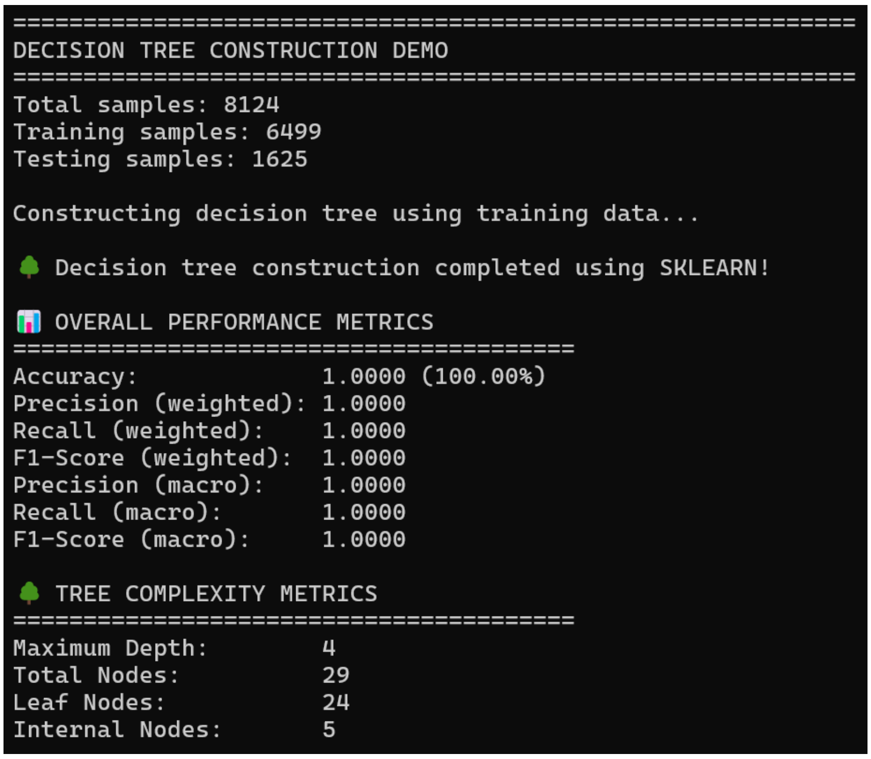
Name: Nandita R Nadig **SRN:** PES2UG23CS365

Outputs:

a) mushroom.csv

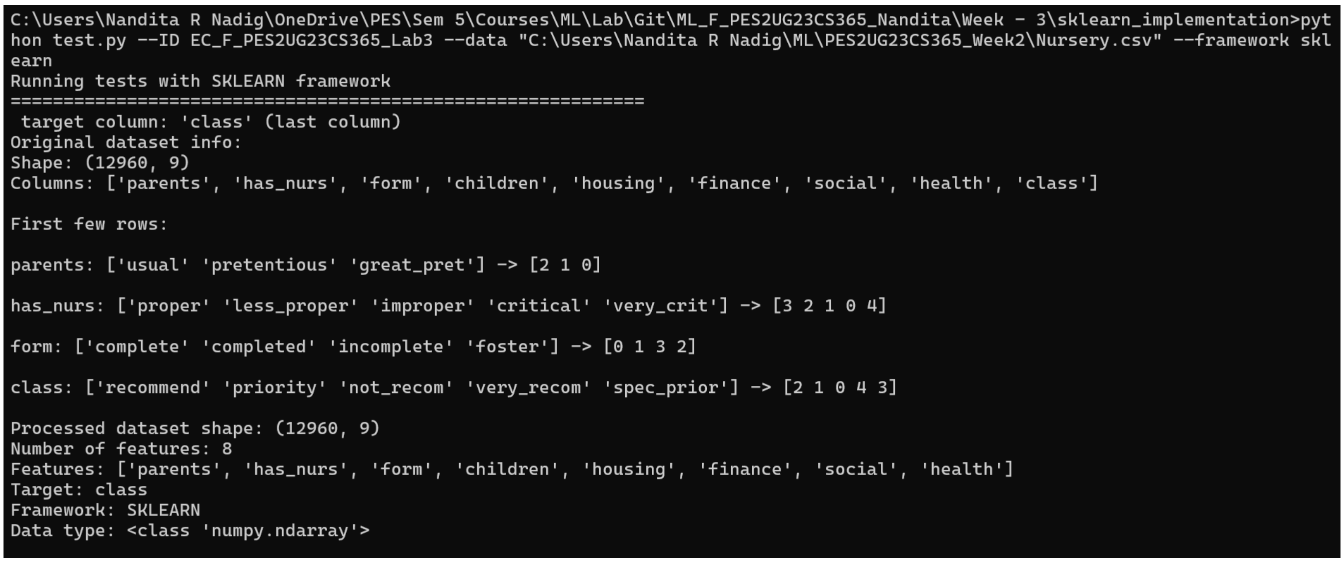


```
C:\Users\Nandita R Nadig\OneDrive\PES\Sem 5\Courses\ML\Lab\Git\ML_F_PES2UG23CS365_Nandita\Week - 3\sklearn_implementation>pyt
hon test.py --ID EC_F_PES2UG23CS365_Lab3 --data "C:\Users\Nandita R Nadig\ML\PES2UG23CS365_Week2\mushrooms.csv" --framework s
klearn
Running tests with SKLEARN framework
============================================================
 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-c
olor', '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', 'clas
s']

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

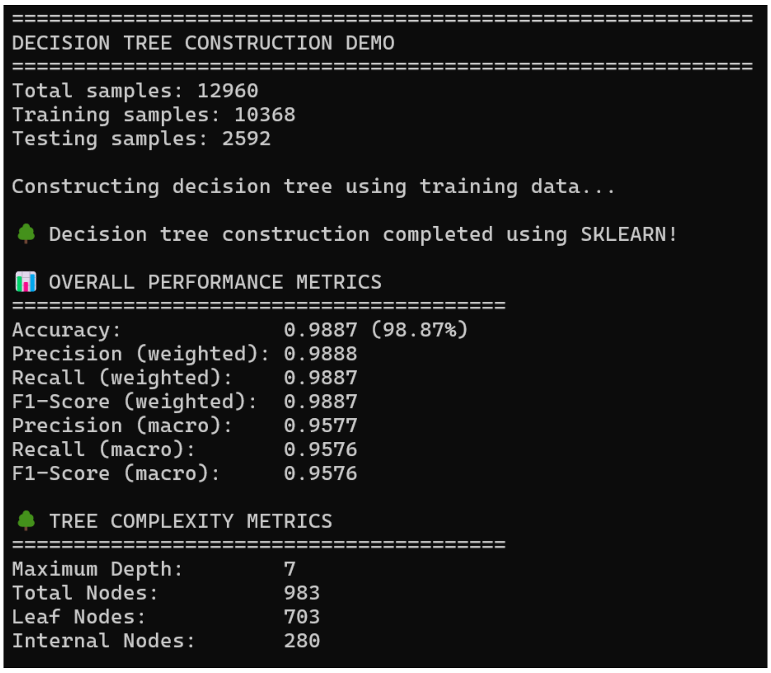


```
============================================================
DECISION TREE CONSTRUCTION DEMO
============================================================
Total samples: 8124
Training samples: 6499
Testing samples: 1625

Constructing decision tree using training data...

🌳 Decision tree construction completed using SKLEARN!

📊 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
```

b) Nursery.csv

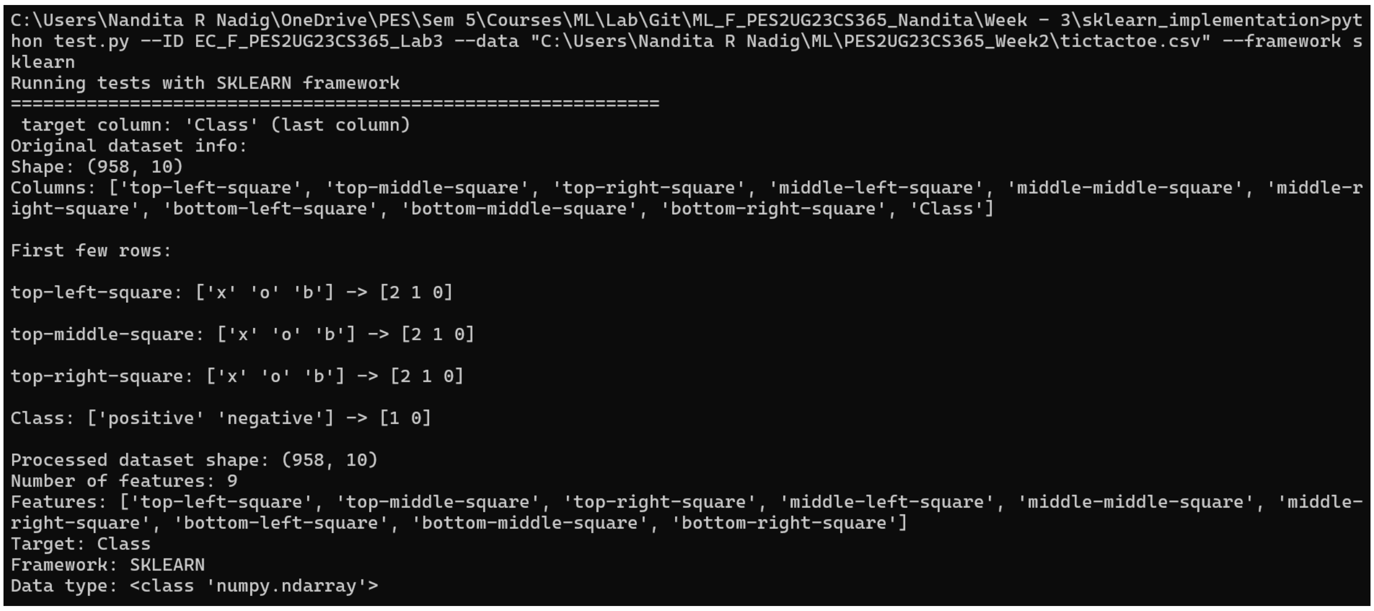


```
C:\Users\Nandita R Nadig\OneDrive\PES\Sem 5\Courses\ML\Lab\Git\ML_F_PES2UG23CS365_Nandita\Week - 3\sklearn_implementation>pyt
hon test.py --ID EC_F_PES2UG23CS365_Lab3 --data "C:\Users\Nandita R Nadig\ML\PES2UG23CS365_Week2\Nursery.csv" --framework skl
earn
Running tests with SKLEARN framework
============================================================
 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'>
```

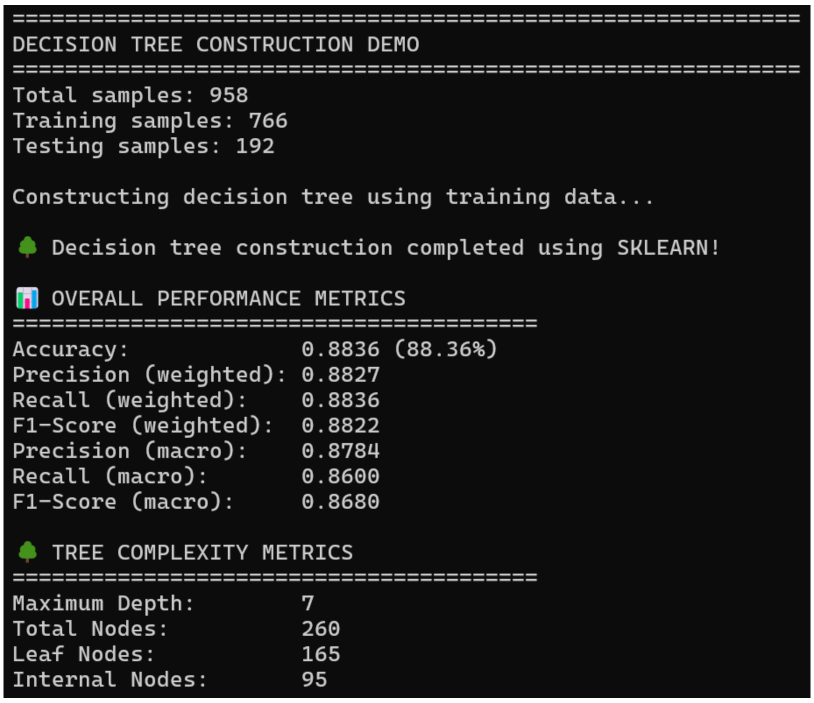


```
============================================================
DECISION TREE CONSTRUCTION DEMO
============================================================
Total samples: 12960
Training samples: 10368
Testing samples: 2592

Constructing decision tree using training data...

🌳 Decision tree construction completed using SKLEARN!

📊 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
```

c) tictactoe.csv



```
C:\Users\Nandita R Nadig\OneDrive\PES\Sem 5\Courses\ML\Lab\Git\ML_F_PES2UG23CS365_Nandita\Week - 3\sklearn_implementation>pyt
hon test.py --ID EC_F_PES2UG23CS365_Lab3 --data "C:\Users\Nandita R Nadig\ML\PES2UG23CS365_Week2\tictactoe.csv" --framework s
klearn
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-r
ight-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!

📊 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):        0.8600
F1-Score (macro):      0.8680

🌳 TREE COMPLEXITY METRICS
========================================
Maximum Depth:         7
Total Nodes:           260
Leaf Nodes:            165
Internal Nodes:        95
```

1) Performance Comparison

Dataset	Accuracy	Precision (weighted)	Recall (weighted)	F1 score (weighted)	Precision (macro)	Recall (macro)	F1 score (macro)
Tic-Tac-Toe	0.8836	~0.888	~0.884	~0.882	-	0.8604	0.8600
Nursery	0.9887	0.9888	0.9888	0.9887	-	0.9576	0.9576
Mushrooms	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Insights:

- The Mushrooms dataset gave a perfect result (100%) because just one or two features like odor are enough to clearly tell apart edible and poisonous.
- The Nursery dataset performed very well (~99%), but since some classes had fewer examples, the model was slightly less accurate for those.
- The Tic-Tac-Toe dataset was the toughest (~88%) because the board positions (x, o, blank) don't give strong signals on their own, so the tree had to grow big and still made some mistakes.

2) Tree Characteristics Analysis

Dataset	Max depth	Total nodes	Leaf nodes	Internal nodes
Tic-Tac-Toe	7	260	165	95
Nursery	7	793	390	403
Mushrooms	≈9	49	24	25

<u>*Insights:*</u>

- The Mushrooms dataset gave a perfect result (100%) because just one or two features like odor are enough to clearly tell apart edible and poisonous.
- The Nursery dataset performed well but since some classes had fewer examples, the model was slightly less accurate for those.
- The Tic-Tac-Toe dataset was also fine but comparatively lower because the board positions (x, o, blank) don't give strong signals on their own, so the tree had to grow big and still made some mistakes.

3) Dataset – Specific Insights

A) Mushrooms (edible vs poisonous)

- *Feature importance:* The most deciding factor is odor, with a couple of gill and spore traits adding support.
- *Class distribution:* Classes are balanced, and the features separate them perfectly.
- *Decision patterns:* Very short rules, like *odor=foul → poisonous*.
- *Overfitting signs:* None – the tree is small, yet accuracy is 100%, showing the dataset is naturally easy to split.

B) Nursery (5 classes)

- *Feature importance:* Features like finance, parents, social, and health dominate the top of the tree, while others like housing and children refine the results.
- *Class distribution:* Skewed toward the not_recom class, which makes the model slightly weaker on the smaller classes (macro F1 lower than weighted).
- *Decision patterns:* The tree creates clear rules, e.g., *poor finance → not_recom unless good social/health factors are present*.
- *Overfitting signs:* Large number of nodes shows the tree is complex. Still, accuracy is very high and generalizes well.

C) Tic-Tac-Toe (positive vs negative outcome)

- *Feature importance:* The center and corners of the board matter most for deciding the result.
- *Class distribution:* More "negative" than "positive" games, so the model struggles more with the smaller class.
- *Decision patterns:* Longer and very specific paths, like *center=x → corner=x → win/block square*, leading to many nodes
- *Overfitting signs:* Tree is quite large, suggesting it could benefit from pruning or depth limits.

4) Comparative Analysis Report

a) Algorithm Performance:

a. Which dataset achieved the highest accuracy and why?

Ans: The Mushrooms dataset gave a perfect score (100%). This is because one attribute, odor, almost fully decides whether a mushroom is edible or poisonous. The rest of the features just handle smaller cases, so the classes are separated very clearly.

b. How does dataset size affect performance?

Ans: Having more data does not always lead to better accuracy. For example, the Nursery dataset is the largest among those considered and performs well. However, the Mushrooms dataset, despite being smaller, achieves even better performance due to its more meaningful and informative features. This shows that while dataset size can matter, the quality and relevance of features often play a more important role in determining model accuracy.

c. What role does the number of features play?

Ans: The number of features alone doesn't determine performance; it depends on how informative those features are. A dataset with a few strong features can achieve high accuracy with a simple tree. In contrast, weaker features make the tree grow larger and perform less effectively. In multi-class problems, even with good features, the tree can still become large because it needs to manage many possible outcomes.

b) Data Characteristics Impact:

a. How does class imbalance affect tree construction?

Ans: When the data is imbalanced, the decision tree naturally leans toward the majority class, because splitting in that direction reduces error more quickly. This can make the tree less sensitive to the minority class, leading to uneven branch growth where majority-class paths are shorter and minority-class paths are deeper or less accurate. This might result in drop of recall and F1 score for the smaller class.

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

Ans: Binary features work best when they are strongly linked to the target (like *odor* in the Mushroom dataset) because they lead to simple and highly accurate trees. Multi-valued or weak features make the tree larger and more complex, since the model has to create many branches to cover all possibilities, which can reduce efficiency and sometimes accuracy.

c) Practical Applications:

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

Ans:

- Tic-Tac-Toe → Useful for game strategy evaluation, teaching how machines "learn rules," or exploring patterns in sequential moves.
- Nursery → Fits decision-support systems like school admissions or recommendation systems where multiple factors decide an outcome.
- Mushrooms → Directly relevant for food safety, screening edible vs poisonous mushrooms using a few clear traits.

b. What are the interpretability advantages for each domain?

Ans:

- Tic-Tac-Toe → Trees show “if-then” game rules, easy to explain to learners.
- Nursery → Rules can be translated into policy guidelines, keeping decisions transparent and auditable.
- Mushrooms → Very small, clear rules (“odor = foul → poisonous”), making safety checks highly interpretable.

c. How would you improve performance for each dataset?

Ans:

- Tic-Tac-Toe → Simplify the tree with pruning or use ensembles (Random Forest) to capture complex patterns better.
- Nursery → Handle class imbalance (resampling or weights), calibrate probabilities, and prune for simpler rules for better clarity.
- Mushrooms → Can add safety nets like rejecting uncertain inputs or validating with external checks.