ML Lab 3
Name: Dinakar Emmanuel
SRN: PES2UG23CS178
Class: 5C

Q1: Compare the following metrics across all three datasets.

Metric →	Accuracy	Precision		Recall		F1-Score	
		Weighted	Macro	Weighted	Macro	Weighted	Macro
Mushrooms	1	1	1	1	1	1	1
Nursery	0.9887	0.9888	0.9577	0.9887	0.9576	0.9887	0.9576
TicTacToe	0.8836	0.8827	0.8784	0.8836	0.8600	0.8822	0.8680

Q2: Tree Characteristics Analysis

Characteristics →	Depth	Nodes			No. of features
		Total	Leaf	Internal	
Mushrooms	4	29	24	5	22
Nursery	7	983	703	280	8
TicTacToe	7	260	165	95	9

Characteristics →	Root	Early splits
Mushrooms	‘odor’	‘spore-print-color’, ‘habitat’
Nursery	‘health’	‘has_nurs’, ‘parents’
TicTacToe	‘middle-middle-square’	‘bottom-left-square’, ‘top-right-square’

Relationship between tree size and dataset characteristics:

- Greater depth does not always mean higher accuracy. Dataset complexity, class separability, and number of features matter more.

- A higher number of features (Mushrooms, 22) doesn't necessarily mean a larger tree. If the features are highly discriminative, the tree remains shallow and small.
- Fewer features (Nursery, TicTacToe) can lead to larger trees, since the model needs more splits to separate classes.
- Datasets with more overlap or complex class boundaries result in trees growing bigger with more leaves to capture variations.

Q3: Dataset specific insights

- Which attribute contributes most to the classification?
    - The attribute at the root of the tree.
- Signs of overfitting:
    - Large depth of the tree
    - Too many nodes compared to number of features
    - Some sub-trees are deeper than others.

Q4: Comparative analysis report

- Performance
    - Q: Which dataset achieved highest accuracy, why?
      A: Mushrooms, more distinguishing features results in a shallow and wide tree with high accuracy.
    - Q: How does dataset size affect performance?
      A: small dataset size - risk of overfitting, larger - better generalisation, too large - accuracy may not improve much
    - Q: What role does the number of features play?
      A: few - low accuracy, more - high accuracy up to a point, too many - risk of overfitting
- Data characteristics impact
    - Q: How does class imbalance affect tree construction?
      A: bias towards major classes, weaker splits for minor classes, uneven tree growth, increased depth for subtrees of minor classes.
    - Q: Which types of features (binary vs multi-valued) work better?
      A: binary - may have deeper trees, accuracy depends on combination of features; multi-valued - may achieve high accuracy with less depth with discriminative features since a split can have many child nodes.
- Practical applications
    - Q: For which real-world scenarios is each dataset type most relevant?
      A: mushrooms - food safety, quality control, toxicology;
      nursery - complex, decision-making systems with different priorities.
      tictactoe - game AI, and strategy

$ python test.py --ID EC_C_PES2UG23CS178_Lab3 --data mushrooms.csv --framework sklearn --print-tree

```
/content# python test.py --ID EC_C_PES2UG23CS178_Lab3 --data mushrooms.csv --framework sklearn --print-tree
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', 'g
ill-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: (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-rin
g', '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!

🌲 DECISION TREE STRUCTURE
============================================================
Root [odor] (gain: 0.9048)
├── = 0:
│   ├── Class 0
├── = 1:
│   ├── Class 1
├── = 2:
│   ├── Class 1
├── = 3:
│   ├── Class 0
├── = 4:
│   ├── Class 1
├── = 5:
```

```
│   ├── [spore-print-color] (gain: 0.1487)
│   ├── = 0:
│   │   ├── Class 0
│   ├── = 1:
│   │   ├── Class 0
│   ├── = 2:
│   │   ├── Class 0
│   ├── = 3:
│   │   ├── Class 0
│   ├── = 4:
│   │   ├── Class 0
│   ├── = 5:
│   │   ├── Class 1
│   ├── = 7:
│   │   ├── [habitat] (gain: 0.2767)
│   │   ├── = 0:
│   │   │   ├── [gill-size] (gain: 0.6374)
│   │   │   ├── = 0:
│   │   │   │   ├── Class 0
│   │   │   └── = 1:
│   │   │       ├── Class 1
│   │   ├── = 1:
│   │   │   ├── Class 0
│   │   ├── = 2:
│   │   │   ├── [cap-color] (gain: 0.8267)
│   │   │   ├── = 1:
│   │   │   │   ├── Class 0
│   │   │   ├── = 4:
```

```
│   │   │   ├── = 8:
│   │   │   │   ├── Class 1
│   │   │   └── = 9:
│   │   │       ├── Class 1
│   │   ├── = 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
/content#
```

$ python test.py --ID EC_C_PES2UG23CS178_Lab3 --data Nursery.csv --framework sklearn

```
/content# python test.py --ID EC_C_PES2UG23CS178_Lab3 --data Nursery.csv --framework sklearn
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
```

$ python test.py --ID EC_C_PES2UG23CS178_Lab3 --data tictactoe.csv --framework sklearn

```
/content# python test.py --ID EC_C_PES2UG23CS178_Lab3 --data tictactoe.csv --framework sklearn
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', 'mid
dle-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', 'mi
ddle-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
```