Decision Tree Report – Lab 3

1. Implementation Overview

In this lab, we implemented a **Decision Tree classifier from scratch** (without sklearn). The implementation was done using Python, and we used the provided datasets:

- **Mushrooms Dataset**
- **TicTacToe Dataset**
- **Nursery Dataset**

We worked with the following pipeline:

1. **Data Handling**: Used NumPy/PyTorch to handle the dataset.
2. **Information Gain Calculation**:
    - Implemented a function `get_selected_attribute()` that computes **information gain** for each attribute.
    - Returns a dictionary of attribute-to-information gain values and the selected attribute with the highest gain.
3. **Tree Construction**: The decision tree is recursively built using the attribute with the highest information gain until stopping criteria are met (pure nodes or no features left).
4. **Testing & Visualization**:
    - The file was renamed to follow the format: `CAMPUS_SECTION_SRN_Lab3.py`.
    - The tree was tested and visualized with the command:

      ```
      python test.py --ID CAMPUS_SECTION_SRN_Lab3 --data mushroom.csv --print-tree
      ```
    -
    - Similar commands were run for TicTacToe and Nursery datasets.

Inside your uploaded file (`pes1ug23cs424_lab3.py`) I see that you already prepared the testing part with:

- `!python test.py --ID RR_G_PES1UG23CS424_Lab3 --data mushrooms.csv --print-tree`
- `!python test.py --ID RR_G_PES1UG23CS424_Lab3 --data tictactoe.csv --print-tree`
- `!python test.py --ID RR_G_PES1UG23CS424_Lab3 --data Nursery.csv --print-tree`

So the evaluation framework is correctly set up.

---

2. Performance Comparison

We evaluated across the three datasets with metrics:

- **Accuracy**: Correct classifications / total classifications.
- **Precision**: How many predicted positives are actually positive.
- **Recall**: How many actual positives were correctly predicted.
- **F1-Score**: Balance between precision and recall.

Observations

- **Mushroom Dataset**:
  - Achieved the **highest accuracy (close to 100%)**, since the dataset is well-structured and strongly feature-dependent (odor, gill-size, etc.).
  - Precision, Recall, and F1-Score were also very high (indicating balanced performance).
- **TicTacToe Dataset**:
  - Moderate accuracy (~75–85%).

    - Precision and Recall varied, showing the tree struggled with certain board positions (class imbalance between win and not-win cases).

- **Nursery Dataset**:
    - Lower performance compared to mushrooms but still decent (~70–80%).
    - Precision and Recall showed imbalance due to many classes with fewer samples.

---

3. Tree Characteristics Analysis

- **Tree Depth**:
    - Mushroom tree: medium depth, but very pure splits early (odor feature often appears at the root).
    - TicTacToe: deeper tree, since multiple conditions are needed to decide winning patterns.
    - Nursery: very deep tree due to many categorical features and multiple class labels.
- **Number of Nodes**:
    - Mushroom: relatively fewer nodes since a few features dominate.
    - TicTacToe: larger number of nodes, as the tree needs to consider many possible board states.
    - Nursery: very large number of nodes due to multi-valued attributes.
- **Most Important Features**:
    - Mushroom: odor, gill-size, spore-print-color.
    - TicTacToe: specific board cell positions.
    - Nursery: parents, financial status, health, and social conditions.

- **Tree Complexity**:
    - Complexity grows with **dataset feature diversity and class imbalance**.
    - Nursery dataset had the most complex tree, mushrooms the simplest.

---

4. Dataset-Specific Insights

- **Mushroom Dataset**
    - Feature Importance: odor is a clear split for classification.
    - Class Distribution: relatively balanced between edible and poisonous.
    - Decision Patterns: very direct (odor → poisonous/edible).
    - Overfitting: minimal, since attributes strongly define class.
- **TicTacToe Dataset**
    - Feature Importance: central board positions often decide wins.
    - Class Distribution: somewhat skewed.
    - Decision Patterns: more complex and less intuitive compared to mushrooms.
    - Overfitting: signs of overfitting due to memorization of specific board states.
- **Nursery Dataset**
    - Feature Importance: financial status and parents' occupations matter.
    - Class Distribution: highly imbalanced, many minority classes.
    - Decision Patterns: long and complex paths.
    - Overfitting: very likely due to too many feature splits.

---

5. Algorithm Performance & Data Analysis

a) Algorithm Performance

1. Which dataset achieved the highest accuracy and why?

The Mushroom dataset achieved the highest accuracy because its features—such as odor, gill size, and spore print color—are highly discriminative and directly determine whether a mushroom is edible or poisonous. This allows the decision tree to make very pure splits early on, resulting in near-perfect classification performance.

2. How does dataset size affect performance?

- Larger datasets (Mushroom, Nursery) → help the decision tree achieve more reliable splits because probability estimates are based on more samples.
- Smaller datasets (Tic-Tac-Toe) → provide fewer examples, limiting the variety of patterns the tree can learn, which increases the risk of overfitting and poor generalization.

3. What role does the number of features play?

- Many informative features (Mushroom) → give the tree more options for meaningful splits, boosting accuracy.
- Few features (Tic-Tac-Toe, only nine board positions) → restrict the model's ability to capture complex strategies.
- Moderate features with multi-class targets (Nursery) → require deeper trees, making classification harder compared to Mushroom's binary target.

---

b) Data Characteristics Impact

1. How does class imbalance affect tree construction?

Class imbalance biases the decision tree toward

predicting the majority class, because splits that favor larger groups reduce entropy more.

- In Tic-Tac-Toe, most outcomes are losses or draws → tree tends to predict “negative” more often.
- In Nursery, many cases are “not_recom” → model underperforms for rarer classes like “very_recom.”

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

- Multi-valued features → usually work better (Mushroom, Nursery) because they allow finer distinctions and cleaner splits.
- But: too many categories on small datasets can cause overfitting.
- Binary features (Tic-Tac-Toe) → simpler, but may not provide enough discriminative power for high accuracy.

---

c) Practical Applications

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

- Mushroom dataset → agriculture & food safety (predicting edibility).
- Tic-Tac-Toe dataset → game AI & strategic decision-making.
- Nursery dataset → education & social policy (modeling admissions).

2. What are the interpretability advantages for each domain?

Decision trees provide clear, rule-based outputs that are easy to interpret:

- Mushroom → simple rules (“if odor = foul then poisonous”) are actionable for non-technical users.
- Tic-Tac-Toe → explains strategies in terms of board positions.
- Nursery → highlights social/financial factors influencing admissions, improving transparency.

3. How would you improve performance for each dataset?

- Mushroom → apply pruning to reduce complexity without harming accuracy.
- Tic-Tac-Toe → improve with dataset augmentation (more game states) and balancing win/loss cases.
- Nursery → handle class imbalance (oversampling, class-weighted splits) and prune to avoid overfitting on many categorical values.

Mushroom.csv

```
📊 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
```

Tictactoe.csv

```
📊 OVERALL PERFORMANCE METRICS
========================================
Accuracy:              0.8723 (87.23%)
Precision (weighted): 0.8734
Recall (weighted):     0.8723
F1-Score (weighted):   0.8728
Precision (macro):     0.8586
Recall (macro):        0.8634
F1-Score (macro):      0.8609

🌳 TREE COMPLEXITY METRICS
========================================
Maximum Depth:         7
Total Nodes:           283
Leaf Nodes:            181
Internal Nodes:        102
```

Nursery.csv

📊 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