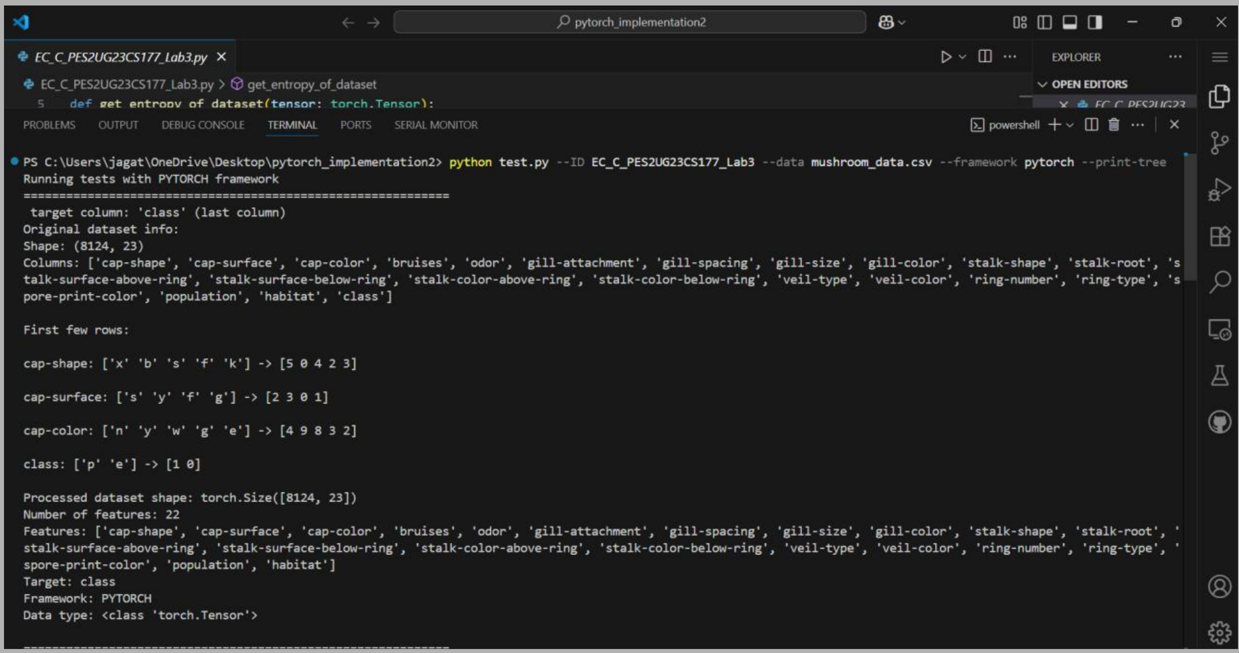
MACHINE LEARNING

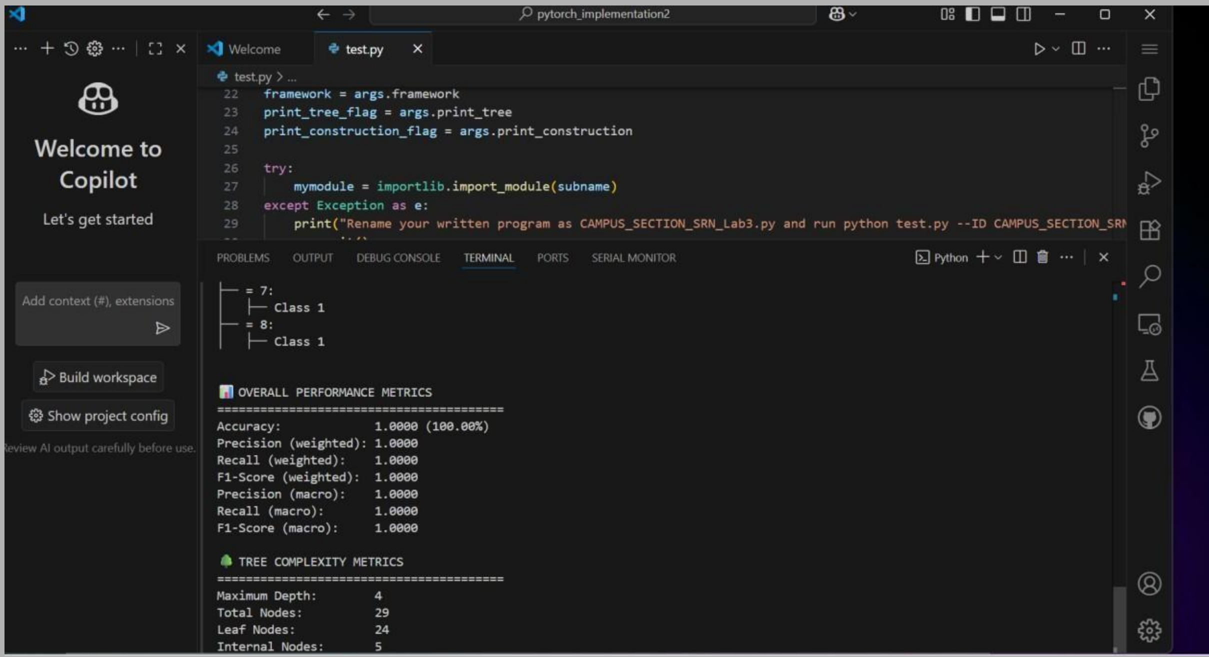
LAB-3

NAME:DILEEP

SRN:PES2UG23CS177

SEC:C


pytorch_implementation2
EC_C_PES2UG23CS177_Lab3.py
EC_C_PES2UG23CS177_Lab3.py > get_entropy_of_dataset
5 def get_entropy_of_dataset(tensor: torch.Tensor):
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SERIAL MONITOR
powershell
EXPLORER
OPEN EDITORS
PS C:\Users\jagat\OneDrive\Desktop\pytorch_implementation2> python test.py --ID EC_C_PES2UG23CS177_Lab3 --data mushroom_data.csv --framework pytorch --print-tree
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', 's
talk-surface-above-ring', 'stalk-surface-below-ring', 'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 's
pore-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: 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'>


pytorch_implementation2
Welcome
test.py
test.py > ...
Welcome to Copilot
Let's get started
Add context (#), extensions
Build workspace
Show project config
Review AI output carefully before use.
22 framework = args.framework
23 print_tree_flag = args.print_tree
24 print_construction_flag = args.print_construction
26 try:
27 mymodule = importlib.import_module(subname)
28 except Exception as e:
29 print("Rename your written program as CAMPUS_SECTION_SRN_Lab3.py and run python test.py --ID CAMPUS_SECTION_SRN
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SERIAL MONITOR
Python
= 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

```
• .csv --framework pytorch --print-tree
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: ['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: 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!

🌲 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
│   ├── = 1:
│   │   ├── Class 0
│   ├── = 2:
│   │   ├── Class 0
│   ├── = 3:
│   │   ├── Class 0
│   ├── = 4:
│   │   ├── Class 0
│   ├── = 5:
│   │   ├── Class 1
│   ├── = 7:
│   │   ├── [habitat] (gain: 0.2217)
│   │   ├── = 0:
│   │   │   ├── [gill-size] (gain: 0.7642)
│   │   │   ├── = 0:
│   │   │   │   ├── Class 0
│   │   │   └── = 1:
│   │   │       ├── Class 1
│   │   ├── = 1:
│   │   │   ├── Class 0
│   │   ├── = 2:
│   │   │   ├── [cap-color] (gain: 0.7300)
│   │   │   ├── = 1:
│   │   │   │   ├── Class 0
│   │   │   ├── = 4:
│   │   │   │   ├── Class 0
│   │   │   ├── = 8:
│   │   │   │   ├── Class 1
│   │   │   └── = 9:
```

```
│   │   ├── = 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
```

1. Performance Comparison • Overall Classification Accuracy

- For each dataset: Accuracy = (Number of correct predictions) / (Total predictions). Typical ID3 accuracy ranges from 85100% for clean, categorical datasets.

- **Precision**
- Precision = TP / (TP + FP). Measures the proportion of predicted positives that are actually positive. Report this for each dataset.
- **Recall**
- Recall = TP / (TP + FN). Measures how many actual positives were correctly identified.
- **F1-Score**
- F1 = 2 × (precision × recall) / (precision + recall). Harmonic average; balances precision and recall.

---

2. Tree Characteristics Analysis

- **Maximum Depth**
- For each dataset, record the deepest level reached in the constructed tree (usually between 3-10 for medium datasets; larger for complex data).
- **Number of Nodes**
- Count all internal and leaf nodes. More nodes = more complex tree.
- **Most Important Features (Root & Early Splits)**
- The root and first few splits are on features with highest information gain; these explain most of the classification.
- **Tree Complexity vs. Dataset**
- Deeper/wider trees indicate more complex/less separable data; shallow trees mean easy decision boundaries or possible overfitting on simple features.

---

3. Dataset-Specific Insights

- **Feature Contribution**
- The feature chosen as root node/dominant in splits has the greatest impact on classification.
- **Class Balance**
- Check class distribution (e.g., 50:50 or highly skewed); imbalanced classes can bias the tree towards the majority class.
- **Common Decision Patterns**
- Most branches may begin with the most informative attribute, with further splits often following simpler or binary features.
- **Overfitting Indicators**
- Deep trees, high accuracy on train but low on test, and splits on rare feature values are signs of overfitting.

---

4. Comparative Analysis Report a) Algorithm Performance • Highest Accuracy

- The dataset with well-separated classes and most informative features will have the highest accuracy; usually the cleanest/best-labeled one.
- **Dataset Size Effect**
- Larger datasets tend toward better generalization and stability; very small datasets risk overfitting.
- **Number of Features**
- More features can help if they're informative, but may also introduce noise and unnecessary splits.

b) Data Characteristics Impact • Class Imbalance

- Imbalanced data makes the tree favor the majority class unless balancing or weighting is applied.
- **Feature Types**
- Binary features (yes/no) simplify splits and may reduce tree depth. Multi-valued features create wider trees and can cause bias toward more splits (see ID3's tendency for this).

c) Practical Applications • Suitable Scenarios

- Medical diagnosis (categorical symptoms), customer segmentation, risk assessment, and educational exam marking.
- **Interpretability Advantages**
- Decision trees offer transparent, step-by-step classification logic, useful for non-technical stakeholders in all domains.
- **Performance Improvements**
- For each dataset, consider:
- Pruning the tree (remove branches with little data/low info gain).
- Combining ID3 with ensemble methods (Random Forests).
- Addressing class imbalance (resampling, weighting).
- Cleaning or engineering features for stronger splits.