PES UNIVERSITY

CELEBRATING 50 YEARS

Dept. of CSE

UE23CS352A: Machine Learning

LAB 3: Decision Tree Classifier - Multi-Dataset Analysis

Dataset Descriptions

Dataset 1: Mushroom Classification

Classification Task: Predict whether a mushroom is edible or poisonous based on its physical characteristics.
Target Classes: Edible (e) → 0, Poisonous (p) → 1

Features: Categorical attributes describing mushroom characteristics including cap-shape, cap-surface, cap-color, bruises, odor, gill properties, stalk properties, veil properties, ring properties, spore-print-color, population, and habitat.

Target: 'class' (last column)

Dataset 2: Tic-Tac-Toe Endgame

Classification Task: Predict the outcome of a Tic-Tac-Toe game (win/loss) based on the current board configuration.
Target Classes: Positive (win) → 1, Negative (loss/draw) → 0

Features: 9 categorical attributes representing board positions (top-left, top-middle, top-right, middle-left, middle-middle, middle-right, bottom-left, bottom-middle, bottom-right). Each position can be 'x', 'o', or 'b' (blank).

Target: 'Class' (positive=1, negative=0)

Dataset 3: Nursery School

Classification Task: Predict the recommendation level for nursery school admission based on family and social factors.
Target Classes: 5 classes - recommend=2, priority=1, not_recom=0, very_recom=4, spec_prior=3
Features: 8 categorical attributes describing family circumstances including parents, has_nurs, form, children, housing, finance, social, and health.
Target: 'class'

Task Overview

Objective

Implement the ID3 Decision Tree algorithm and perform comparative analysis across three diverse datasets to understand algorithm performance under different data characteristics.

Files Provided

1. lab_boilerplate.py - Contains function skeletons for implementation
2. test.py - Testing framework with evaluation metrics
3. Dataset files: mushroom.csv, tictactoe.csv, nursery.csv

lab_boilerplate.py

Functions to Implement

Function Name	Input	Output
get_entropy_of_dataset	PyTorch: tensor: torch.Tensor NumPy: data: np.ndarray representing the given dataset	dataset_entropy: int/float, entropy of the entire dataset
get_avg_info_of_attribute	PyTorch: 1. tensor: torch.Tensor NumPy: 1. data: np.ndarray 2. attribute: int, number representing the attribute	avg_info: int/float, average Information of that attribute
get_information_gain	PyTorch: 1. tensor: torch.Tensor NumPy: 1. data: np.ndarray 2. attribute: int, number representing the attribute	information_gain: int/float, information gain of that attribute

Function Name	Input	Output
get_selected_attribute	PyTorch: tensor: torch.Tensor NumPy: data: np.ndarray representing the given dataset	Result: tuple(information_gains, selected_attribute) where - information_gains: python dictionary with key as attribute number and value as its information gain - selected_attribute: int, attribute number of chosen attribute

test.py

1. This will help you check your code.
2. Rename Decision Tree.py file to CAMPUS_SECTION_SRN_Lab3.py

Testing and Visualization

Basic Testing:

python test.py --ID CAMPUS_SECTION_SRN_Lab3 --data mushroom.csv

python test.py --ID CAMPUS_SECTION_SRN_Lab3 --data tictactoe.csv

python test.py --ID CAMPUS_SECTION_SRN_Lab3 --data nursery.csv

Tree Visualization:

python test.py --ID CAMPUS_SECTION_SRN_Lab3 --data mushroom.csv --print-tree

For sklearn implementations :

bash

python test.py --ID CAMPUS_SECTION_SRN_Lab3 --data mushroom.csv --framework sklearn

Analysis Requirements

1. Performance Comparison

Compare the followingmetricsacrossallthreedatasets:

•Accuracy: Overall classification accuracy

•Precision: True positives / (True positives + False positives)

•Recall: True positives / (True positives + False negatives

•F1-Score: Harmonic mean of precision and recall

2.Tree Characteristics Analysis

Analyze and compare:

• Tree Depth: Maximum depth of the constructed trees

• Number of Nodes: Total nodes in each tree

• Most Important Features: Attributes selected as root and early splits

• Tree Complexity: Relationship between tree size and dataset characteristics

3. Dataset-Specific Insights

For each dataset, analyze:

• Feature Importance: Which attributes contribute most to classification

• Class Distribution: How balanced are the target classes

• Decision Patterns: Common decision paths in the tree

• Overfitting Indicators: Signs of overfitting in tree structure

4. Comparative Analysis Report

Write a comprehensive report addressing:

a) Algorithm Performance:

   a. • Which dataset achieved the highest accuracy and why?
   b. • How does dataset size affect performance?
   c. • What role does the number of features play?

b)Data Characteristics Impact: • How does class imbalance affect tree construction?
•Which types of features (binary vs multi-valued) work better?
c)Practical Applications: • For which real-world scenarios is each dataset type most relevant?
•What are the interpretability advantages for each domain?

- How would you improveperformanceforeachdataset?

Implementation Guidelines

Important Requirements

1. No Hardcoding: Functions must work with any dataset structure
2. Framework Options: You may use either PyTorch or NumPy for implementation
    1. PyTorch: All tensor operations must use PyTorch tensors
    2. NumPy: All array operations must use NumPy arrays
    3. Sklearn library calls are NOT allowed
3. Target Variable: Always assume the last column contains the target variable
4. Function Signatures: Do not modify provided function definitions
5. Additional Functions: You may create helper functions as needed