import math

from collections import Counter

# Example dataset: Outlook, Temperature, Humidity, Wind, Play

dataset = [

['Sunny', 'Hot', 'High', 'Weak', 'No'],

['Sunny', 'Hot', 'High', 'Strong', 'No'],

['Overcast', 'Hot', 'High', 'Weak', 'Yes'],

['Rain', 'Mild', 'High', 'Weak', 'Yes'],

['Rain', 'Cool', 'Normal', 'Weak', 'Yes'],

['Rain', 'Cool', 'Normal', 'Strong', 'No'],

['Overcast', 'Cool', 'Normal', 'Strong', 'Yes'],

['Sunny', 'Mild', 'High', 'Weak', 'No'],

['Sunny', 'Cool', 'Normal', 'Weak', 'Yes'],

['Rain', 'Mild', 'Normal', 'Weak', 'Yes'],

['Sunny', 'Mild', 'Normal', 'Strong', 'Yes'],

['Overcast', 'Mild', 'High', 'Strong', 'Yes'],

['Overcast', 'Hot', 'Normal', 'Weak', 'Yes'],

['Rain', 'Mild', 'High', 'Strong', 'No']

]

features = ['Outlook', 'Temp', 'Humidity', 'Wind']

# Calculate entropy of a dataset

def entropy(data):

labels = [row[-1] for row in data]

counts = Counter(labels)

total = len(data)

return -sum((count/total) \* math.log2(count/total) for count in counts.values())

# Calculate information gain for a feature

def info\_gain(data, feature\_index):

total\_entropy = entropy(data)

values = set(row[feature\_index] for row in data)

subsets = [[row for row in data if row[feature\_index] == val] for val in values]

weighted\_entropy = sum((len(subset)/len(data)) \* entropy(subset) for subset in subsets)

return total\_entropy - weighted\_entropy

# Find the best feature to split

def best\_feature(data, features):

gains = [info\_gain(data, i) for i in range(len(features))]

return gains.index(max(gains))

# Build decision tree recursively

def id3(data, features):

labels = [row[-1] for row in data]

if labels.count(labels[0]) == len(labels):

return labels[0]

if not features:

return Counter(labels).most\_common(1)[0][0]

best = best\_feature(data, features)

tree = {features[best]: {}}

values = set(row[best] for row in data)

for val in values:

subset = [row[:best] + row[best+1:] for row in data if row[best] == val]

new\_features = features[:best] + features[best+1:]

tree[features[best]][val] = id3(subset, new\_features)

return tree

# Function to classify a new sample

def classify(tree, features, sample):

if not isinstance(tree, dict):

return tree

root = next(iter(tree))

value = sample[features.index(root)]

branch = tree[root].get(value)

if branch is None:

return "Unknown"

new\_features = [f for f in features if f != root]

return classify(branch, new\_features, sample)

# Build the decision tree

decision\_tree = id3(dataset, features)

print("Decision Tree:", decision\_tree)

# Test classification

sample = ['Sunny', 'Cool', 'Normal', 'Strong'] # New case

print("Classification for", sample, ":", classify(decision\_tree, features, sample))