Exp-15 implement decision tree

Code

import math

def entropy(labels):

counts = {}

for label in labels:

counts[label] = counts.get(label, 0) + 1

ent = 0

total = len(labels)

for count in counts.values():

p = count / total

ent -= p \* math.log2(p)

return ent

def best\_feature\_to\_split(data):

base\_entropy = entropy([row[-1] for row in data])

best\_gain = 0

best\_feature = 0

num\_features = len(data[0]) - 1

for i in range(num\_features):

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

new\_entropy = 0

for v in values:

subset = [row for row in data if row[i] == v]

p = len(subset) / len(data)

new\_entropy += p \* entropy([row[-1] for row in subset])

info\_gain = base\_entropy - new\_entropy

if info\_gain > best\_gain:

best\_gain = info\_gain

best\_feature = i

return best\_feature

def build\_tree(data, features):

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

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

return labels[0]

if len(data[0]) == 1:

return max(set(labels), key=labels.count)

best = best\_feature\_to\_split(data)

tree = {features[best]: {}}

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

for v in values:

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

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

tree[features[best]][v] = build\_tree(subset, sub\_features)

return tree

data = [

['Sunny', 'Hot', 'No'],

['Sunny', 'Cool', 'Yes'],

['Rain', 'Hot', 'Yes'],

['Rain', 'Cool', 'No'],

]

features = ['Outlook', 'Temperature']

tree = build\_tree(data, features)

print(tree)

output

