***ERICK MORALES  
https://github.com/erijmo/3690  
ID3***  
  
import pandas as pd

import math

import copy

data = {

'Toothed': ['Toothed', 'Toothed', 'Toothed', 'Not Toothed', 'Toothed', 'Toothed', 'Toothed', 'Toothed', 'Toothed', 'Not Toothed'],

'Hair': ['Hair', 'Hair', 'Not Hair', 'Hair', 'Hair', 'Hair', 'Not Hair', 'Not Hair', 'Not Hair', 'Not Hair'],

'Breathes': ['Breathes', 'Breathes', 'Breathes', 'Breathes', 'Breathes', 'Breathes', 'Not Breathes', 'Breathes', 'Breathes', 'Breathes'],

'Legs': ['Legs', 'Legs', 'Not Legs', 'Legs', 'Legs', 'Legs', 'Not Legs', 'Not Legs', 'Legs', 'Legs'],

'Species': ['Mammal', 'Mammal', 'Reptile', 'Mammal', 'Mammal', 'Mammal', 'Reptile', 'Reptile', 'Mammal', 'Reptile']

}

attribute = ['Toothed', 'Hair', 'Breathes', 'Legs']

class Node(object):

def \_\_init\_\_(self):

self.value = None

self.decision = None

self.childs = None

def find\_entropy(data, rows):

yes = sum(1 for i in rows if data['Species'][i] == 'Mammal')

no = sum(1 for i in rows if data['Species'][i] == 'Reptile')

entropy = 0

if yes != 0 and no != 0:

x = yes / (yes + no)

y = no / (yes + no)

entropy = -1 \* (x \* math.log2(x) + y \* math.log2(y))

return entropy

def find\_max\_gain(data, rows, columns):

max\_gain = 0

ret\_idx = -1

entropy = find\_entropy(data, rows)

if entropy == 0:

return max\_gain, ret\_idx

for j in columns:

my\_dict = {}

idx = j

for i in rows:

key = data[attribute[idx]][i]

if key not in my\_dict:

my\_dict[key] = 1

else:

my\_dict[key] += 1

gain = entropy

for key in my\_dict:

yes = sum(1 for k in rows if data['Species'][k] == 'Mammal' and data[attribute[idx]][k] == key)

no = sum(1 for k in rows if data['Species'][k] == 'Reptile' and data[attribute[idx]][k] == key)

x = yes / (yes + no)

y = no / (yes + no)

if x != 0 and y != 0:

gain += (my\_dict[key] \* (x \* math.log2(x) + y \* math.log2(y))) / len(rows)

if gain > max\_gain:

max\_gain = gain

ret\_idx = j

return max\_gain, ret\_idx

def build\_tree(data, rows, columns):

max\_gain, idx = find\_max\_gain(data, rows, columns)

root = Node()

root.childs = []

if max\_gain == 0:

root.value = 'Mammal' if sum(1 for i in rows if data['Species'][i] == 'Mammal') > 0 else 'Reptile'

return root

root.value = attribute[idx]

my\_dict = {}

for i in rows:

key = data[attribute[idx]][i]

if key not in my\_dict:

my\_dict[key] = 1

else:

my\_dict[key] += 1

new\_columns = copy.deepcopy(columns)

new\_columns.remove(idx)

for key in my\_dict:

new\_rows = [i for i in rows if data[attribute[idx]][i] == key]

temp = build\_tree(data, new\_rows, new\_columns)

temp.decision = key

root.childs.append(temp)

return root

def traverse(root):

print(root.decision)

print(root.value)

n = len(root.childs)

if n > 0:

for i in range(0, n):

traverse(root.childs[i])

def predict\_new\_example(new\_example, root):

current\_node = root

while current\_node.childs:

attribute\_value = new\_example[current\_node.value]

child\_node = None

for child in current\_node.childs:

if child.decision == attribute\_value:

child\_node = child

break

if child\_node:

current\_node = child\_node

else:

break

return current\_node.value

def calculate():

rows = [i for i in range(len(data['Toothed']))]

columns = [i for i in range(len(attribute))]

root = build\_tree(data, rows, columns)

root.decision = 'Start'

new\_example = {

'Toothed': 'Toothed',

'Hair': 'Hair',

'Breathes': 'Not Breathes',

'Legs': 'Legs'

}

predicted\_species = predict\_new\_example(new\_example, root)

print(f"Predicted species: {predicted\_species}")

calculate()

**OUTPUT**

Predicted species: Mammal