**Experiment-3:**

**Write a program to demonstrate the working of decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.**

**#Implementation**

import pandas as pd

import math

importnumpy as np

data = pd.read\_csv("/content/PlayTennis.csv")

features = [feat for feat in data]

features.remove("answer")

#Create a class named Node with four members children, value, isLeaf and pred.

class Node:

    def \_\_init\_\_(self):

        self.children = []

        self.value = ""

        self.isLeaf = False

        self.pred = ""

#Define a function called entropy to find the entropy oof the dataset.

def entropy(examples):

    pos = 0.0

    neg = 0.0

    for \_, row in examples.iterrows():

        if row["answer"] == "yes":

            pos += 1

        else:

            neg += 1

    ifpos == 0.0 or neg == 0.0:

        return 0.0

    else:

        p = pos / (pos + neg)

        n = neg / (pos + neg)

        return -(p \* math.log(p, 2) + n \* math.log(n, 2))

#Define a function named info\_gain to find the gain of the attribute

definfo\_gain(examples, attr):

    uniq = np.unique(examples[attr])

    #print ("\n",uniq)

    gain = entropy(examples)

    #print ("\n",gain)

    for u in uniq:

        subdata = examples[examples[attr] == u]

        #print ("\n",subdata)

        sub\_e = entropy(subdata)

        gain -= (float(len(subdata)) / float(len(examples))) \* sub\_e

        #print ("\n",gain)

    return gain

#Define a function named ID3 to get the decision tree for the given dataset

def ID3(examples, attrs):

    root = Node()

    max\_gain = 0

    max\_feat = ""

    for feature in attrs:

        #print ("\n",examples)

        gain = info\_gain(examples, feature)

        if gain >max\_gain:

            max\_gain = gain

            max\_feat = feature

    root.value = max\_feat

    #print ("\nMax feature attr",max\_feat)

    uniq = np.unique(examples[max\_feat])

    #print ("\n",uniq)

    for u in uniq:

        #print ("\n",u)

        subdata = examples[examples[max\_feat] == u]

        #print ("\n",subdata)

        if entropy(subdata) == 0.0:

            newNode = Node()

            newNode.isLeaf = True

            newNode.value = u

            newNode.pred = np.unique(subdata["answer"])

            root.children.append(newNode)

        else:

            dummyNode = Node()

            dummyNode.value = u

            new\_attrs = attrs.copy()

            new\_attrs.remove(max\_feat)

            child = ID3(subdata, new\_attrs)

            dummyNode.children.append(child)

            root.children.append(dummyNode)

    return root

#Define a function named printTree to draw the decision tree

defprintTree(root: Node, depth=0):

    fori in range(depth):

        print("\t", end="")

    print(root.value, end="")

    ifroot.isLeaf:

        print(" -> ", root.pred)

    print()

    for child in root.children:

        printTree(child, depth + 1)

#Define a function named classify to classify the new example

def classify(root: Node, new):

    for child in root.children:

        ifchild.value == new[root.value]:

            ifchild.isLeaf:

                print ("Predicted Label for new example", new," is:", child.pred)

                exit

            else:

                classify (child.children[0], new)

#Finally, call the ID3, printTree and classify functions

root = ID3(data, features)

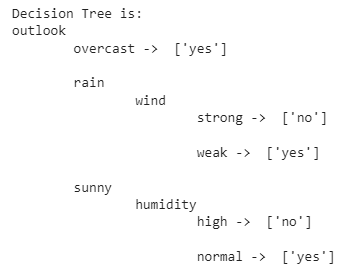
print("Decision Tree is:")

printTree(root)

print ("------------------")

new = {"outlook":"sunny", "temperature":"hot", "humidity":"normal", "wind":"strong"}

classify (root, new)



==============

Predicted Label for new example {'outlook': 'sunny', 'temperature': 'hot', 'humidity': 'normal', 'wind': 'strong'} is: ['yes']