**Name: YATHIN K V**

**USN: 1PE17IS105**

**Class: ISE 7B**

**Machine Learning Laboratory**

Program 3

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

**ID3 Algorithm :**

1. Create a Root node for the tree.

2. If all Examples are positive, Return the single node tree Root, with label=+

3. If all Examples are negative, Return the single node tree Root, with label=-

4. If Attributes is empty, Retun the sinle node tree Root, with label=most common value of TargetAttribute in Examples

5. Otherwise Begin

• A<- the attribute from Attributes that best classifies Examples

• The decision attribute for Root <-A

• For each possible value, vi, of A

▪ Add a new branch below Root, corresponding to the test A=vi

▪ Let Examplesvi be the subset of Examples that have value vi for A

▪ If Examplesvi is empty

- Then below this new branch add a leaf node with label=most common value of TargetAttribute in Examples

- Else below this new branch add the subtree

ID3(Examplesvi , TargetAttribute, Attributes- {A})

6. End

7. Return Root

**PROGRAM:**

import pandas as pd

import math

class Node:

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

self.label=l

self.branches = {}

def entropy(data):

total\_ex = len(data)

positive\_ex = len(data.loc[data["Play Tennis"] == 'Y'])

negative\_ex = len(data.loc[data["Play Tennis"] == 'N'])

entropy = 0

if(positive\_ex > 0):

entropy = (-1)\*(positive\_ex/float(total\_ex))\*(math.log(positive\_ex,2)-math.log(total\_ex,2))

if(negative\_ex > 0):

entropy += (-1)\*(negative\_ex/float(total\_ex))\*(math.log(negative\_ex,2)-math.log(total\_ex,2))

return entropy

def gain(s,data,attrib):

values = set(data[attrib])

print(values)

gain = s

for val in values:

gain -= len(data.loc[data[attrib] == val])/float(len(data))\*entropy(data.loc[data[attrib] == val])

return gain

def get\_attrib(data):

entropy\_s = entropy(data)

attribute =""

max\_gain = 0

for attr in data.columns[:len(data.columns)-1]:

g = gain(entropy\_s,data,attr)

if g > max\_gain:

max\_gain = g

attribute = attr

return attribute

def decision\_tree(data):

root = Node("NULL")

if(entropy(data) == 0):

if(len(data.loc[data[data.columns[-1]] == 'Y']) == len(data)):

root.label = "Y"

return root

else:

root.label = "N"

return root

if(len(data.columns) == 1):

return

else:

attrib = get\_attrib(data)

root.label = attrib

values = set(data[attrib])

for val in values:

root.branches[val] = decision\_tree(data.loc[data[attrib] == val].drop(attrib,axis = 1))

return root

def get\_rules(root,rule,rules):

if not root.branches:

rules.append(rule[:-2]+" => "+root.label)

return rules

for i in root.branches:

get\_rules(root.branches[i],rule+root.label+"="+i+" ^ ",rules)

return rules

def test(tree,test\_str):

if not tree.branches:

return tree.label

return test(tree.branches[test\_str[tree.label]],test\_str)

data = pd.read\_csv('Data3.csv')

entropy\_s = entropy(data)

attrib\_count = 0

cols = len(data.columns)-1

tree = decision\_tree(data)

rules = get\_rules(tree,"",[])

print(rules)

test\_str = {}

print("Enter test case input")

for i in data.columns[:-1]:

test\_str[i] = input(i+": ")

print(test\_str)

print(test(tree,test\_str))

**OUTPUT:**





**CSV FILE:**

Outlook,Temperature,Humidity,Windy,Play Tennis

Sunny,Hot,High,Weak,N

Sunny,Hot,High,Strong,N

Overcast,Hot,High,Weak,Y

Rain,Mild,High,Weak,Y

Rain,Cool,Normal,Weak,Y

Rain,Cool,Normal,Strong,N

Overcast,Cool,Normal,Strong,Y

Sunny,Mild,High,Weak,N

Sunny,Cool,Normal,Weak,Y

Rain,Mild,Normal,Weak,Y

Sunny,Mild,Normal,Strong,Y

Overcast,Mild,High,Strong,Y

Overcast,Hot,Normal,Weak,Y

Rain,Mild,High,Strong,N