Name: Daksh Sharma Roll No: 101615039 Group: ENC2

# Experiment 4

**Aim**: Write a program to implement Decision tree using ID3.

## Program:

#true-strong false-weak

import numpy as np;

import copy;

outlook= ["sunny","sunny","overcast","rainy","rainy","rainy","overcast","sunny","sunny","rainy","sunny","overcast","overcast","rainy"]

temp= ['hot','hot','hot','mild','cool','cool','cool','mild','cool','mild','mild','mild','hot','mild']

humidity=['high','high','high','high','normal','normal','normal','high','normal','normal','normal','high','normal','high']

wind= ['false','true','false','false','false','true','true','false','false','false','true','true','false','true']

play=["no","no","yes","yes","yes","no","yes","no","yes","yes","yes","yes","yes","no"]

training\_data=[outlook,temp,humidity,wind,play]

def maptovaluesofattri(attribute):

nout=['sunny','overcast','rainy']

nt=['hot','mild','cool']

nhum=['high','normal']

nwind=['false','true']

if(attribute=='outlook'):

return nout

elif(attribute=='temp'):

return nt

elif(attribute=='humidity'):

return nhum

else:

return nwind

def selectindexattri(attribute):

if(attribute=='outlook'):

return 0

elif(attribute=='temp'):

return 1

elif(attribute=='humidity'):

return 2

elif(attribute=='wind'):

return 3

else:

return 4

def entropy(training\_data,z,atv): #atv-attribute value sunny overcast rainy etc

noyes=0

nono=0

lena=0

if(atv=='' and z==''):

for i in range(0,len(training\_data[4])):

if(training\_data[4][i]=='yes'):

noyes+=1

lena+=1

elif(training\_data[4][i]=='no'):

nono+=1

lena+=1

else:

for i in range(0,len(training\_data[z])):

if(training\_data[z][i]==atv and training\_data[4][i]=='yes'):

noyes+=1

lena+=1

elif(training\_data[z][i]==atv and training\_data[4][i]=='no'):

nono+=1

lena+=1

if(lena==noyes or lena==nono):

return 0

elif(noyes==nono):

return 1

else:

return -(noyes/lena)\*np.log2(noyes/lena)-(nono/lena)\*np.log2(nono/lena)

def info\_gain(training\_data,attribute,parent\_entropy):

labels={}

ig=parent\_entropy

z=selectindexattri(attribute)

for label in training\_data[z]:

if(label not in labels.keys()):

labels[label]=0

labels[label]+=1

for lab in labels:

ig-=(labels[lab]/len(training\_data[z]))\*entropy(training\_data,z,lab)

return ig

def recursive\_split(training\_data,rem\_labels):

tree={}

if(len(rem\_labels)==0):

return None

else:

max\_ig=0

parent\_entropy=entropy(training\_data,'','')

if(parent\_entropy==0):

return training\_data[4][0]

else:

for i in rem\_labels:

x=info\_gain(training\_data,i,parent\_entropy)

if(x>max\_ig):

max\_ig=x

val=i

for x in range(0,len(rem\_labels)):

if(rem\_labels[x] == val):

break

rem\_labels.pop(x)

y=maptovaluesofattri(val)

z=selectindexattri(val)

tree['node\_val']=val

for i in range(0,len(y)): ## sunny overcast rainy

datasubset=copy.deepcopy(training\_data)

j=0

while(j<len(datasubset[z])):

if(datasubset[z][j]!=y[i]):

for k in range(0,len(datasubset)):

datasubset[k].pop(j)

else:

j+=1

tree[y[i]]=recursive\_split(datasubset,rem\_labels)

return tree

### main ###

tree={}

S=entropy(training\_data,'','') #root entropy

rem\_labels=['outlook','temp','humidity','wind']

max\_ig=0

val=""

for i in rem\_labels:

x=info\_gain(training\_data,i,S)

if(x>max\_ig):

max\_ig=x

val=i

for x in range(0,len(rem\_labels)):

if rem\_labels[x] == val:

break

rem\_labels.pop(x)

y=maptovaluesofattri(val)

z=selectindexattri(val)

tree['node\_val']=val

k=0

for i in range(0,len(y)): ## sunny overcast rainy

datasubset=copy.deepcopy(training\_data)

j=0

while(j<len(datasubset[z])):

if(datasubset[z][j]!=y[i]):

for k in range(0,len(datasubset)):

datasubset[k].pop(j)

else:

j+=1

tree[y[i]]=recursive\_split(datasubset,rem\_labels)

print('DECISIONTREE:\n',tree,'\n\nPOSSIBLE INPUTS:','\noutlook:',maptovaluesofattri('outlook'),'\ntemp:',maptovaluesofattri('temp'),'\nhumidity:',maptovaluesofattri('humidity'),'\nwind:',maptovaluesofattri('wind'))

loop=1

while(loop):

inp=input('Enter input for prediction-- [outlook,temp,humidity,wind]:')

inp=eval(inp)

ans=copy.deepcopy(tree)

while(ans!='yes' or ans!='no' or ans!=None):

if(ans['node\_val']=='outlook'):

ans=ans[inp[0]]

if(ans=='yes' or ans=='no'):

break

elif(ans['node\_val']=='temp'):

ans=ans[inp[1]]

if(ans=='yes' or ans=='no'):

break

elif(ans['node\_val']=='humidity'):

ans=ans[inp[2]]

if(ans=='yes' or ans=='no'):

break

elif(ans['node\_val']=='wind'):

ans=ans[inp[3]]

if(ans=='yes' or ans=='no'):

break

print('Prediction:',ans)

loop=input('Enter 1 to make prediction again 0 to exit:')

## Results:

In this dataset decision tree perfectly classifies the training data but overfitting might take place in dataset which is otherwise, there we will have to set a stopping criteria or allow the trees to fully grow and then prune the branches.

