16	
03	Hoste 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.
	import pandar as pd
	impost numpy as np
	import math
	data = pd. read_csv("dodaset.csv")
	features = [f for f in data]
	features. remove ("answer")
	records of the contract of the
	class Node:
	def -init - (relf):
	Selfachildren=[]
	self-value = ""
	self. Prleaf = Falre
	Self pord=""
	defentropy (examples):
	DO1 = 0.0
	그 없는 그리다 그리다는 가 없는 아이들은 그리고 있다. 그리고 아이들은 그리고 있는 그리고 있는 그리고 있다.
	for , row in examples iterrous():
	if row ["answer"]== "yes":
	나는 사람들이 아니는 사람들이 되었다면 하는 것이 없는 것이 없는 것이 없는 것이 없는 것이 없는 것이 없는 것이 없다면 없는 것이다. 그렇게 되었다면 없는 것이 없는 것이 없다면 없는 것이다. 그렇게 되었다면 없는 것이 없는 것이 없다면 없는 것이다면 없다면 없는 것이다면 없다면 없다면 없다면 없다면 없다면 없다면 없다면 없다면 없다면 없
	else:
	보냈다는 그리 이번 모양이 하루 보다 그리는 경기를 받아 있다. 나는 사람이 그는 아니라인 그리고 하는 사람들은 모양이 불어나는 사람이다.
	neg +=1
	00 == 000
0.7	if pos == 00 ox neg == 00;
	return 0.0
	else:
	p = pos / (pos + neg)
(A) and a	n=neg/(postneg)

return - (p x math elog (p,2) + n x mat	h.
(og(n,2))	
det Entogain (examples, atts)?	
unig = npo unique (examples [att x])	
gain = entrophy (examples)	
for uning:	
Subdata = examples [examples [attr	7 == u)
Sub-e = entrophy (subdata)	J
gain=(float (len (subdata))/float (Jen (
examples)) X Sub-e	
return gain	
	4
def ID3 (examples, attog):	ă.
root = Node()	
	Ž.
max-gain=0	
max-feat = " "	
for feature in attre:	
gain = into gain (example, feature)	
if gain; max gain;	
max gain = gain	
rocx feat = feature	
root. value: max-feature	
0	_
unig = np. unique (examples[max feat])
for u in unique	
subdata = example: [example, I mex	feat
it en tonne (contrate)	and the second
if (ntropy (subdata)== 0.0: newNode = Node()	
newNoellilleaf = True	
new Node value = u	

	newNocle, pred = np. unique (subdata ["anewer"])
	root children append (newNode)
	else:
	dummy Node = Node()
	dumny Node. value = 11
	dummy Node value = u
	new-attologemore (max feat)
	child = ID3 (subdata, new-attre)
	dummywode children append (child)
	root hildren append (dunny Node)
	return root
24 (1) 1 1 1 1	
	les point Tree (voot: Node, depth=0):
	= for i in range (depth):
	point ("\t", end = "")
	print (root-value, end = "")
	proved (real for the)
	if root. illeaf:
	print (" =)", root, pred)
	print()
	for child in root, children:
	pointTree (child, depth +1)
,	root = ID3 (data, features)
	printTree (root)
	F. W. C. 1 . C. C