Harsh RJah 13417csozi

program; import numpy as up import math from dala-loader import read-data class Nede, def -int- (self, altributi):
eff.atibuti =altributi self. children = 17 self com=" dy -sh-bill? rehern self, altribute. def sublables (dala, col): dict = [] ifen = np. cenique (data[:, 101]) count-inp. jeroes ((item. shapes 10], 1), dtype-int) for x in varge (itens: shape (0)): forey in varye (data shope 107): if data by, woll == itens In) counter] += 1

fr a in range (item hope 107):

dict (item (xi)) = np. empty clint (earnt (last)), data. shall

other e = '532')

pos=0

for y in range (data: shape 107):

TITLE:		
water + control (at al phantal Hall the	Expt. No. :	
	Date :	
of data (y, wi) === items [m]:	Hanh R	
diel (ikus (x)) /pos7: dala	sy) flansh R IBG17 (30)	
fues += 1		
dict (items (x)) = np. delut (dict [items 1, col, 1)		
return items, dict		
led enhopy (1):		
items = np. unique (s)		
d de maria de la companya del companya de la companya del companya de la companya		
rehenn o		
seems=0 ((item shape 10), 1)	1)	
for n in varge (thous shape (0)): courts / n) - quan (5 == item (n)) / ((5.2.2)	
has egget in county.		
sunsted rounds math. log (went, 2)		
gaineratio (dala, w));		
stem, otich = sublables (data, col)		
fokul_sign = data. shap (0)		
fokul-sife = data. shape (0) entropies = rp. jenous ((ikms-shape 107,1))	Marks:	
S	Staff:	

for x in vary (ifems shape 10):

ratio = dict (items 1807) shape 107 / (to tal-sit)

ontropies (v7 = valo + entropy (dicts itensha)) total-entropy = entropy (datos:, -1)) for a in vary (enhopres - shape 101)): hobed-entropy -= entropies (2) setena botal-enhapy def auat mode (dala, nederdata): if Enpremique (data (:,-17)). shap to] == 1? node = Node ("") node answer = np. unique (data 1:, 7)) (0) return node. garles expresses ((dalorshape 17-1,1)) for col iv varge colate shape (17-1):
gains [w] = gain_raho (dato, col) speil- np. argmas (galus) mode= readi(metaddata (spbit)) metadata = pp. delete (mbudata, split, '0') items, dict= sublables (data, speit) for x in varge (items shape 10); child = create - node (dict (item 127), metodata) noch. children append (Citas 12), child ! suher node.

TITLE:	
	Expt. No. :
	Date:
alif empty (sije):	Harsh Mach
S= " "	134170031
for nin range (sije) St: "	
St = " "	
outern s.	
des print-tree (node, level):	
y node. answer! = ""	
point (amply (level), nod annuer)	
return.	
print Compty (level), node altributi)	
for value, a in node children:	
print compty (end), value)	
portur were (n, lud +2)	
metadata, trandata e read-dala cutrain	ing example esse)
data = np. avray (man date)	
nodi: creati mode (data, meladata)	
print-tree (node, o)	
	Marks:
	Stoff .

Output 3. Stay
Rary
b'No'
Surmy
b'Yes'

PROGRAM 3

By Harsh R Jain 1BG17CS031

```
import numpy as np
import math
from data_loader import read_data
class Node:
    def __init__(self, attribute):
        self.attribute = attribute
        self.children = []
        self.answer = ""
    def __str__(self):
        return self.attribute
def subtables(data, col):
    dict = {}
    #unique values of a particular attribute
    items = np.unique(data[:, col])
    #initializes the count of an attribute value in the training data to zero
    count = np.zeros((items.shape[0], 1), dtype=int)
    #counts the no. of occurance of an attribute value in the training data
    for x in range(items.shape[0]):
        for y in range(data.shape[\emptyset]):
            if data[y, col] == items[x]:
                count[x] += 1
    for x in range(items.shape[0]):
        dict[items[x]] = np.empty((int(count[x]), data.shape[1]),
dtype="S32")
        pos = \emptyset
        #create a dict containing key as the attribute value and value as the
list of instances with attribute value
        for y in range(data.shape[0]):
            if data[y, col] == items[x]:
                dict[items[x]][pos] = data[y]
                pos += 1
```

```
#from the dict created above remove the value which matches with the
dict key
        dict[items[x]] = np.delete(dict[items[x]], col, 1)
   return items, dict
def entropy(S):
   #the no. of target attribute values
    items = np.unique(S)
   #if the collection contains only 1 element the entropy value is zero
   if items.size == 1:
        return 0
   #initializes the count of instances with the target attribute
values(yes/no) to zero
    counts = np.zeros((items.shape[0], 1))
   sums = 0
   #proportion of positive and negative instances
    for x in range(items.shape[0]):
        counts[x] = sum(S == items[x]) / (S.size)
    #computing entropy
    for count in counts:
        sums += -1 * count * math.log(count, 2)
    return sums
def gain_ratio(data, col):
    #subtables function returns the possible attribute values and a
dictionary mapping a value to the instances having that value
    items, dict = subtables(data, col )
    total\_size = data.shape[0]
    entropies = np.zeros((items.shape[0], 1))
    #compute info gain of each attribute
    for x in range(items.shape[0]):
        #ratio=count of instances having a attribute value/total no. of
instances
        ratio = dict[items[x]].shape[0]/(total_size)
        entropies[x] = ratio * entropy(dict[items[x]][:, -1])
   total_entropy = entropy(data[:, -1])
```

```
for x in range(entropies.shape[0]):
        total_entropy -= entropies[x]
   return total_entropy
def create_node(data, metadata):
    if (np.unique(data[:, -1])).shape[0] == 1:
        node = Node("")
        node.answer = np.unique(data[:, -1])[0]
        return node
    #gain of each attribute initialized as zero
   gains = np.zeros((data.shape[1] - 1, 1))
   #compute gain of each attribute
    for col in range(data.shape[1] - 1):
        gains[col] = gain_ratio(data, col)
   #index of attribute having maximum gain
   split = np.argmax(gains)
   #attribute having max gain forms a node of the tree
   node = Node(metadata[split])
   #remove the attribute from metadata after making it a node in the tree
   metadata = np.delete(metadata, split, 0)
   #items-possible values of the attribute with max gain, dict- mapping of
each value to instances having that value
    items, dict = subtables(data, split)
   #for each attribute value find the next best attribute
    for x in range(items.shape[0]):
        child = create_node(dict[items[x]], metadata)
        node.children.append((items[x], child))
    return node
def empty(size):
   s = ""
    for x in range(size):
        s += "
```

```
return s
def print_tree(node, level):
          if node.answer != "":
                   print(empty(level), node.answer)
                   return
         print(empty(level), node.attribute)
          for value, n in node.children:
                   print(empty(level + 1), value)
                   print_tree(n, level + 2)
metadata, traindata = read_data("trainingexamples.csv")
data = np.array(traindata)
node = create_node(data, metadata)
print_tree(node, 0)
                                                                                                 Python 3.8.2 Shell
  File Edit Format Run Options Window Help
                                                                                                 File Edit Shell Debug Options Window Help
                                                                                                Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AM D64)] on win32
      rt numpy as np
         math
                                                                                                 Type "help", "copyright", "credits" or "license()" for more information.
   rom data loader import read data
                                                                                                                == RESTART: D:\7th Sem\ML Lab\Program 3\3-ID3\ID3.pv =
      def __init__(self, attribute):
    self.attribute = attribute
    self.children = []
    self.answer = ""
                                                                                                     Rainy
b'No'
      subtables(data, col):
      dict = {)
funique values of a particular attribute
items = np.unique(data[:, col])
finitializes the count of an attribute value in the training data to zero
count = np.zeros((items.shape[0], 1), dtype=int)
foounts the no. of occurance of an attribute value in the training data
for x in range(items.shape[0]):
    for y in range(data.shape[0]):
        if data[y, col] == items[x]:
            count[x] += 1
      for x in range(items.shape[0]):
    dict[items[x]] = np.empty((int(count[x]), data.shape[1]), dtype="S32")
           {\tt pos} = 0 foreate a dict containing key as the attribute value and value as the li
      for y in range(data.shape[0]):
    if data[y, col] == items[x]:
        dict[items[x]][pos] = data[y]
        pos += 1

    #from the dict created above remove the value which matches with the dict dict[items[x]] = np.delete(dict[items[x]], col, 1)
return items, dict
      Lp: 10 Col: 4
                                                                              🧿 刘 🕲 🛱 🖫 📗
```