

PROGRAM 3

```
import
math
import
csv
```

```
def load_csv(filename): lines=csv.reader(open(filename,"r"));
    dataset = list(lines) headers
    = dataset.pop(0)return
    dataset,headers
```

```
class Node:
```

```
    def __init__(self,attribute):
        self.attribute=attribute
        self.children=[]
        self.answer=""
```

```
def subtables(data,col,delete): dic={ }
```

```
    coldata=[row[col] for row in data]
    attr=list(set(coldata))
```

```
    counts=[0]*len(attr)
    r=len(data)
    c=len(data[0])
```

```
    for x in range(len(attr)):for y in
        range(r):
            if data[y][col]==attr[x]:
                counts[x]+=1
```

```
    for x in range(len(attr)):
```

```
        dic[attr[x]]=[[0 for i in range(c)] for j inrange(counts[x])]
```

```
        pos=0
```

```
        for y in range(r):
```

```
            if data[y][col]==attr[x]:if
                delete:
```

```

        del data[y][col]
        dic[attr[x]][pos]=data[y] pos+=1

    return attr,dic

def entropy(S): attr=list(set(S))

    if len(attr)==1:
        return 0

    counts=[0,0]

    for i in range(2):

        counts[i]=sum([1 for x in S if attr[i]==x])/(len(S)*1.0)

    sums=0

    for cnt in counts:

        sums+=-1*cnt*math.log(cnt,2) return

    sums

def compute_gain(data,col): attr,dic = subtables(data,col,delete=False)

    total_size=len(data)
    entropies=[0]*len(attr)
    ratio=[0]*len(attr)

    total_entropy=entropy([row[-1] for row in data])for x in
    range(len(attr)):

        ratio[x]=len(dic[attr[x]])/(total_size*1.0)
        entropies[x]=entropy([row[-1] for row in
dic[attr[x]]])

        total_entropy-=ratio[x]*entropies[x] return

    total_entropy

def build_tree(data,features): lastcol=[row[-1] for row in data]
    if(len(set(lastcol))==1:

        node=Node("")
        node.answer=lastcol[0]
        return node

```

```

n=len(data[0])-
lgains=[0]*n

for col in range(n): gains[col]=compute_gain(data,col)

split=gains.index(max(gains))
node=Node(features[split])

fea = features[:split]+features[split+1:] attr,dic=subtables(data,split,delete=True)

for x in range(len(attr)): child=build_tree(dic[attr[x]],fea)
    node.children.append((attr[x],child))

return node

```

```

def print_tree(node,level): if node.answer!="":
    print("  "*level,node.answer)return

```

```

print("  "*level,node.attribute) for value,n
in node.children:
    print("  "*(level+1),value)
    print_tree(n,level+2)

```

```

def classify(node,x_test,features): if node.answer!="":
    print(node.answer
    )return

pos=features.index(node.attribute) for value,
n in node.children:
    if x_test[pos]==value: classify(n,x_test,features)

```

```

"""Main program""" dataset,features=load_csv("data3.csv")
node1=build_tree(dataset,features)

```

```

print("The decision tree for the dataset using ID3 algorithmis")

print_tree(node1,0)
testdata,features=load_csv("data3_test.csv") for xtest in
testdata:
    print("The test instance:",xtest)

```

```
print("The label for test instance:",end=" ")
classify(node1,xtest,features)
```

OUTPUT:

Decision Tree is:

outlook

overcast -> ['yes']

rain

wind

strong -> ['no']

weak -> ['yes']

sunny

humidity

high -> ['no']

normal -> ['yes']

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