```
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 = "" # NULL indicates children exists.
 # Not Null indicates this is a Leaf Node


def subtables(data, col, delete):
    dic = {}
    coldata = [ row[col] for row in data]
    attr = list(set(coldata)) # All values of attribute retrived

    for k in attr:
       dic[k] = []

    for y in range(len(data)):
       key = data[y][col]
       if delete:
          del data[y][col]
       dic[key].append(data[y])
```

```
    return attr, dic


def entropy(S):
    attr = list(set(S))
    if len(attr) == 1: #if all are +ve/-ve then entropy = 0
        return 0


    counts = [0,0] # Only two values possible 'yes' or 'no'
    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):
    attValues, dic = subtables(data, col, delete=False)


    total_entropy = entropy([row[-1] for row in data])
    for x in range(len(attValues)):
        ratio = len(dic[attValues[x]]) / ( len(data) * 1.0)
        entro = entropy([row[-1] for row in dic[attValues[x]]])
        total_entropy -= ratio*entro


    return total_entropy


def build_tree(data, features):


 lastcol = [row[-1] for row in data]
 if (len(set(lastcol))) == 1: # If all samples have same labels return that label
```

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

n = len(data[0])-1
gains = [compute_gain(data, col) for col in range(n) ]

split = gains.index(max(gains)) # Find max gains and returns index
node = Node(features[split]) # 'node' stores attribute selected
#del (features[split])
fea = features[:split]+features[split+1:]

attr, dic = subtables(data, split, delete=True) # Data will be spilt in subtables
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) # Displays leaf node yes/no
        return

    print(" "*level, node.attribute) # Displays attribute Name
    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") # Read Tennis data
node = build_tree(dataset, features) # Build decision tree

print("The decision tree for the dataset using ID3 algorithm is ")
print_tree(node, 0)

testdata, features = load_csv("data3_test.csv")
for xtest in testdata:
    print("The test instance : ",xtest)
    print("The predicted label : ", end="")
    classify(node,xtest,features)
```