In [1]:

```
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))
    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:
        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):
    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:
        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))
    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)

```

```
datasets,features=load_csv("train_weather1.csv")
node=build_tree(datasets,features)
print("The decision tree for the dataset using ID3 algorithm is")
print_tree(node,0)
testdata,features=load_csv("test_weather1.csv")
for xtest in testdata:
    print("The test instance: ",xtest)
    print("The predicted label: ",end="")
    classify(node,xtest,features)
```

```
The decision tree for the dataset using ID3 algorithm is
 Type
  minivan
   no
  suv
   Tires
    blackwall
     no
    whitewall
     yes
  car
   Doors
    4
     no
    2
     yes
The test instance:  ['blue', 'car', '4', 'blackwall']
The predicted label: no
The test instance:  ['green', 'suv', '4', 'whitewall']
The predicted label: yes
The test instance:  ['red', 'car', '2', 'blackwall']
The predicted label: yes
The test instance:  ['green', 'suv', '2', 'blackwall']
The predicted label: no
The test instance:  ['green', 'minivan', '4', 'whitewall']
The predicted label: no
```

In [2]:

```
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))
    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:
        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):
    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:
        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))
    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)

```

```
datasets,features=load_csv("train_weather.csv")
node=build_tree(datasets,features)
print("The decision tree for the dataset using ID3 algorithm is")
print_tree(node,0)
testdata,features=load_csv("test_weather.csv")
for xtest in testdata:
    print("The test instance: ",xtest)
    print("The predicted label: ",end="")
    classify(node,xtest,features)
```

```
The decision tree for the dataset using ID3 algorithm is
 Outlook
  overcast
   yes
  sunny
   Humidity
    high
     no
    normal
     yes
  rainy
   Wind
    weak
     yes
    strong
     no
The test instance:  ['overcast', 'hot', 'normal', 'weak']
The predicted label: yes
The test instance:  ['overcast', 'cool', 'normal', 'weak']
The predicted label: yes
The test instance:  ['sunny', 'mild', 'high', 'strong']
The predicted label: no
The test instance:  ['undefined', 'hot', 'normal', 'strong']
The predicted label:
```

In [ ]:

```
1
```