# 1.實作ID3 decision tree 二元分類方法

## 第一題

In [1]:
```
import math
def entropy(p1,n1):
    if(p1==0 and n1==0):
        return 1
    elif(p1==0 or n1==0):
        return 0
    else:
        pp=p1/(p1+n1)
        np=n1/(p1+n1)
        return -pp*math.log2(pp)-np*math.log2(np)
# print(entropy(29,35))
```

In [2]:
```
def IG(p1,n1,p2,n2):
    num1 = p1+n1
    num2 = p2+n2
    num=num1+num2
    return entropy(p1+p2,n1+n2)-num1/num*entropy(p1,n1)-num2/num*entropy(p2,n2)
#print(IG(21,5,8,30))
#print(IG(18,33,11,2))
```

In [3]:
```
def ID3DTtrain(feature,target):
    node = dict()
    node['data']=range(len(target))        #data為編號
    tree = []
    tree.append(node)                      #root=node
    t=0
    while(t<len(tree)):                    #編號t要生節點嗎
        idx = tree[t]['data']
        if(sum(target[idx])==0):           #全部的target皆為0
            tree[t]['leaf']=1                #走到底了
            tree[t]['decision']=0            #以後走到這就是0
        elif(sum(target[idx])==len(idx)):  #全部target都是1
            tree[t]['leaf']=1
            tree[t]['decision']=1            #以後走到這都是1
        else:                              #沒切乾淨要再多切幾刀
            bestIG=0
            for i in range(feature.shape[1]):  #對每個feature都看一下
                pool = list(set(feature[idx,i]))  #看看編號idx的node裡的data 它們的feature i都長啥樣
                pool.sort()
                for j in range (len(pool)-1):
                    thres = (pool[j]+pool[j+1])/2 #區分j和j+1的中間值 e.g. 1 7 9 > 4 6(共n-1)
                    G1=[]
                    G2=[]
                    for k in idx:
                        if(feature[k][i]<thres):  #如果小於thres分到group1
                            G1.append(k)
                        else:                     #不是的話就分到 group2
                            G2.append(k)
                    p1=sum(target[G1]==1)
                    n1=sum(target[G1]==0)
                    p2=sum(target[G2]==1)
                    n2=sum(target[G2]==0)
                    thisIG=IG(p1,n1,p2,n2)
                    if(thisIG>bestIG):     #把這個切法記起來
                        bestIG=thisIG
                        bestG1=G1
                        bestG2=G2
                        bestthres=thres
                        bestf = i          #記起來最好的切分的feature是誰
            if(bestIG>0):   #如果切分更細有幫助的話
                tree[t]['leaf']=0
                tree[t]['selectf']=bestf
                tree[t]['threshold']=bestthres
                tree[t]['child']=[len(tree),len(tree)+1]
                node=dict()
                node['data']=bestG1  #小於切分值得放右邊
                tree.append(node)
                node=dict()
                node['data']=bestG2
                tree.append(node)
            else:      #多切無益 姑且挑個比較好的值
                tree[t]['leaf']=1    #走到底了
                if(sum(target[idx]==1)>sum(target[idx]==0)):  #因為1比0多
                    tree[t]['decision']=1    #就讓他當1吧
                else:
                    tree[t]['decision']=0
        t+=1
    return tree

```

In [4]:
```
from sklearn import datasets
data = datasets.load_iris()
feature = data['data']
target = data['target']
T=ID3DTtrain(feature[0:100,:],target[0:100])
```

2.完成單一樣本測試函式

**第二題**

In [6]:
```
def ID3DTtest(Tree,feature1):
    now=0
    while(Tree[now]['leaf']==0):
        bestf = Tree[now]['selectf']
        thres = Tree[now]['threshold']
        if(feature1[bestf]<thres):
            now = Tree[now]['child'][0]
        else:
            now = Tree[now]['child'][1]
    return Tree[now]['decision']
```

In [7]:
```
# print(ID3DTtest(T,feature[130,:]))
```

3. IRIS 資料庫共有 150 筆資料，共有三類花各 50 筆，今僅取後 100 筆（即 target 為 1 及 2 兩 類）來實驗，請用這 100 筆資料建樹，再用相同資料測試，試算其預測正確率。

Ans: 100%

4. 承上，若每類改用前 30 筆建樹，後 20 筆測試，試算其測試資料的預測正確率。

Ans: 95%