**Practical No. 4**

**Aim**: To implement ID3 algorithm

**Prerequisite:**

* Working of ID3 classification algorithm
* Understanding of fundamental programming constructs in C/C++/Java
* Basic features of WEKA tool

**Outcome:** After successful completion of this experiment students will be able to

* Implement the process of selecting the split attribute and analyze its importance in the working of ID3 Algorithm.
* Use Classifier tab in WEKA and create a Tree based classifier model for the data set given and analyze the model created.

**Theory:**

The ID3 algorithm begins with the original set S as the root node. On each iteration of the algorithm, it iterates through every unused attribute of the set S and calculates the [entropy](https://en.wikipedia.org/wiki/Entropy_(information_theory)) H(S) (or [information gain](https://en.wikipedia.org/wiki/Information_gain_in_decision_trees) IG(A)) of that attribute. It then selects the attribute which has the smallest entropy (or largest information gain) value. The set S is then split by the selected attribute (e.g. age is less than 50, age is between 50 and 100, age is greater than 100) to produce subsets of the data. The algorithm continues to recur on each subset, considering only attributes never selected before.  
Recursion on a subset may stop in one of these cases:

* Every element in the subset belongs to the same class (+ or -), then the node is turned into a leaf and labelled with the class of the examples
* There are no more attributes to be selected, but the examples still do not belong to the same class (some are + and some are -), then the node is turned into a leaf and labelled with the most common class of the examples in the subset
* There are no examples in the subset, this happens when no example in the parent set was found to be matching a specific value of the selected attribute, for example if there was no example with age >= 100. Then a leaf is created, and labelled with the most common class of the examples in the parent set.

Throughout the algorithm, the decision tree is constructed with each non-terminal node representing the selected attribute on which the data was split, and terminal nodes representing the class label of the final subset of this branch.

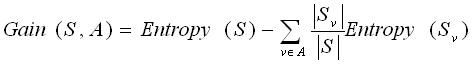
A measure used from Information Theory in the ID3 algorithm and many others used in decision tree construction is that of Entropy. Informally, the entropy of a dataset can be considered to be how disordered it is. It has been shown that entropy is related to information, in the sense that the higher the entropy, or uncertainty, of some data, then the more information is required in order to completely describe that data. In building a decision tree, we aim to decrease the entropy of the dataset until we reach leaf nodes at which point the subset that we are left with is pure, or has zero entropy and represents instances all of one class (all instances have the same value for the target attribute).

We measure the entropy of a dataset,S, with respect to one attribute, in this case the target attribute, with the following calculation:

Entropy Calculation  
where Pi is the proportion of instances in the dataset that take the ith value of the target attribute, which has C different values.

This probability measures give us an indication of how uncertain we are about the data. And we use a log2 measure as this represents how many bits we would need to use in order to specify what the class (value of the target attribute) is of a random instance.

We can use a measure called Information Gain, which calculates the reduction in entropy (Gain in information) that would result on splitting the data on an attribute, A.

  
where v is a value of A , |Sv| is the subset of instances of S where A takes the value v,   
and |S| is the number of instances

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(TO BE COMPLETED BY STUDENTS)

|  |  |
| --- | --- |
| Roll No. K046 | Name: Emil Pulickal |
| Class: B.Tech CSCS 2nd Year Sem 4 | Batch: A2 |
| Date of Practical: 12/02/22 | Date of Submission: 17/02/22 |
| Grade: |  |

1. Implement an ID3 algorithm for selecting the first splitting attribute in the Height data set given below.

Code:

#include <stdio.h>

#include <stdlib.h>

#include<math.h>

struct sample{

int height; //0-1.6 1.6-1.7 1.7-1.8 1.8-1.9 1.9-2.0 2.0-infinity

// 1 2 3 4 5 6

char gender; //f,m 1,2

int output; //s,m,t 1,2,3

}samples[15];

int main()

{

samples[0].height=1;

samples[0].gender=1;

samples[0].output=1;

samples[1].height=5;

samples[1].gender=2;

samples[1].output=3;

samples[2].height=4;

samples[2].gender=1;

samples[2].output=2;

samples[3].height=4;

samples[3].gender=1;

samples[3].output=2;

samples[4].height=2;

samples[4].gender=1;

samples[4].output=1;

samples[5].height=4;

samples[5].gender=2;

samples[5].output=2;

samples[6].height=1;

samples[6].gender=1;

samples[6].output=1;

samples[7].height=2;

samples[7].gender=2;

samples[7].output=1;

samples[8].height=6;

samples[8].gender=2;

samples[8].output=3;

samples[9].height=6;

samples[9].gender=2;

samples[9].output=3;

samples[10].height=3;

samples[10].gender=1;

samples[10].output=2;

samples[11].height=5;

samples[11].gender=2;

samples[11].output=2;

samples[12].height=4;

samples[12].gender=1;

samples[12].output=2;

samples[13].height=3;

samples[13].gender=1;

samples[13].output=2;

samples[14].height=3;

samples[14].gender=1;

samples[14].output=2;

float female=0,male=0;

int block1=0,block2=0,block3=0,block4=0,block5=0,block6=0;

int shortie=0,medium=0,tall=0;

for(int i=0;i<15;i++){

if(samples[i].gender==1){

female+=1;

}

else{male+=1;}

}

for(int i=0;i<15;i++){

if(samples[i].output==1){

shortie+=1;

}

else if(samples[i].output==2){

medium+=1;

}

else if(samples[i].output==3){

tall+=1;

}

}

//printf("F:%d,M:%d",female,male);

float total=shortie+medium+tall;

//S[short,med,tall]

//attribute gender:

float S=-(shortie/total)\*(log2(shortie/total))-(medium/total)\*(log2(medium/total))-(tall/total)\*(log2(tall/total));

float a=0,b=0,c=0,d=0,e=0,f=0;

//male:

for(int i=0;i<15;i++){

if(samples[i].gender==2){

if(samples[i].output==1){

a+=1;

}

else if(samples[i].output==2){

b+=1;

}

else if(samples[i].output==3){

c+=1;

}

}

else if(samples[i].gender==1){

if(samples[i].output==1){

d+=1;

}

else if(samples[i].output==2){

e+=1;

}

else if(samples[i].output==3){

f+=1;

}

}

}

float a1,b1,c1,d1,e1,f1;

float x=a+b+c;

float y=d+e+f;

if(a!=0){

a1=-(a\*((log2(a/x)))/x);

}else{a1=0;}

if(b!=0){

b1=-(b\*((log2(b/x)))/x);

}else{b1=0;}

if(c!=0){

c1=-(c\*((log2(c/x)))/x);

}else{c1=0;}

if(d!=0){

d1=-(d\*((log2(d/y)))/y);

}else{d1=0;}

if(e!=0){

e1=-(e\*((log2(e/y)))/y);

}else{e1=0;}

if(f!=0){

f1=-(f\*((log2(f/y)))/y);

}else{f1=0;}

float val2=d1+e1+f1;

float val1=a1+b1+c1;

//printf("no1 %.3f, no2 %.3f",val1,val2);

//printf("S=%f",S);

float IGgender=S-(male/(male+female))\*val1-(female/(male+female))\*val2;

printf("Info Gain(Gender): %f",IGgender);

//-----------------------------------

//height:

for(int i=0;i<15;i++){

if(samples[i].height==1){

block1+=1;

}

else if(samples[i].height==2){

block2+=1;

}

else if(samples[i].height==3){

block3+=1;

}

else if(samples[i].height==4){

block4+=1;

}

else if(samples[i].height==5){

block5+=1;

}

else if(samples[i].height==6){

block6+=1;

}

}

total=block1+block2+block3+block4+block5+block6;

//attribute height:

//S=-(block1/total)\*(log2(block1/total))-(block2/total)\*(log2(block2/total))-(block3/total)\*(log2(block3/total))-(block4/total)\*(log2(block4/total))-(block5/total)\*(log2(block5/total))-(block6/total)\*(log2(block6/total));

float g=0,h=0,i=0,j=0,k=0,l=0,m=0,n=0,o=0,p=0,q=0,r=0;

a=0,b=0,c=0,d=0,e=0,f=0;

//male:

for(int i=0;i<15;i++){

if(samples[i].height==1){

if(samples[i].output==1){

a+=1;

}

else if(samples[i].output==2){

b+=1;

}

else if(samples[i].output==3){

c+=1;

}

}

else if(samples[i].height==2){

if(samples[i].output==1){

d+=1;

}

else if(samples[i].output==2){

e+=1;

}

else if(samples[i].output==3){

f+=1;

}

}

else if(samples[i].height==3){

if(samples[i].output==1){

g+=1;

}

else if(samples[i].output==2){

h+=1;

}

else if(samples[i].output==3){

i+=1;

}

}

else if(samples[i].height==4){

if(samples[i].output==1){

j+=1;

}

else if(samples[i].output==2){

k+=1;

}

else if(samples[i].output==3){

l+=1;

}

}

else if(samples[i].height==5){

if(samples[i].output==1){

m+=1;

}

else if(samples[i].output==2){

n+=1;

}

else if(samples[i].output==3){

o+=1;

}

}

else if(samples[i].height==6){

if(samples[i].output==1){

p+=1;

}

else if(samples[i].output==2){

q+=1;

}

else if(samples[i].output==3){

r+=1;

}

}

}

//

float g1,h1,i1,j1,k1,l1,m1,n1,o1,p1,q1,r1;

x=a+b+c;

y=d+e+f;

float z=g+h+i;

float u=j+k+l;

float v=m+n+o;

float w=p+q+r;

if(a!=0){

a1=-(a\*((log2(a/x)))/x);

}else{a1=0;}

if(b!=0){

b1=-(b\*((log2(b/x)))/x);

}else{b1=0;}

if(c!=0){

c1=-(c\*((log2(c/x)))/x);

}else{c1=0;}

if(d!=0){

d1=-(d\*((log2(d/y)))/y);

}else{d1=0;}

if(e!=0){

e1=-(e\*((log2(e/y)))/y);

}else{e1=0;}

if(f!=0){

f1=-(f\*((log2(f/y)))/y);

}else{f1=0;}

if(g!=0){

g1=-(g\*((log2(g/z)))/z);

}else{g1=0;}

if(h!=0){

h1=-(h\*((log2(h/z)))/z);

}else{h1=0;}

if(i!=0){

i1=-(i\*((log2(i/z)))/z);

}else{i1=0;}

if(j!=0){

j1=-(j\*((log2(j/u)))/u);

}else{j1=0;}

if(k!=0){

k1=-(k\*((log2(k/u)))/u);

}else{k1=0;}

if(l!=0){

l1=-(l\*((log2(l/u)))/u);

}else{l1=0;}

if(m!=0){

m1=-(m\*((log2(m/v)))/v);

}else{m1=0;}

if(n!=0){

n1=-(n\*((log2(n/v)))/v);

}else{n1=0;}

if(o!=0){

o1=-(o\*((log2(o/v)))/v);

}else{o1=0;}

if(p!=0){

p1=-(p\*((log2(p/w)))/w);

}else{p1=0;}

if(q!=0){

q1=-(q\*((log2(q/w)))/w);

}else{q1=0;}

if(r!=0){

r1=-(r\*((log2(r/w)))/w);

}else{r1=0;}

float val3=a1+b1+c1;

float val4=d1+e1+f1;

float val5=g1+h1+i1;

float val6=j1+k1+l1;

float val7=m1+n1+o1;

float val8=p1+q1+r1;

float IGheight=S-(block1/(total))\*val3-(block2/(total))\*val4-(block3/(total))\*val5-(block4/(total))\*val6-(block5/(total))\*val7-(block6/(total))\*val8;

printf("\nInfo Gain(Height): %f",IGheight);

if(IGheight>IGgender){

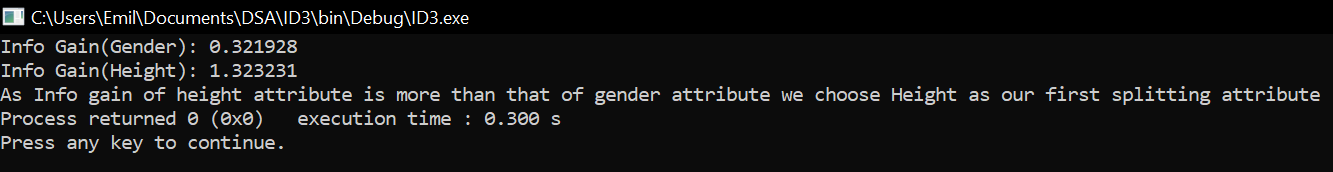
printf("\nAs Info gain of height attribute is more than that of gender attribute we choose Height as our first splitting attribute");}

else{printf("\nAs Info gain of gender attribute is more than that of height attribute we choose Gender as our first splitting attribute");

}

}

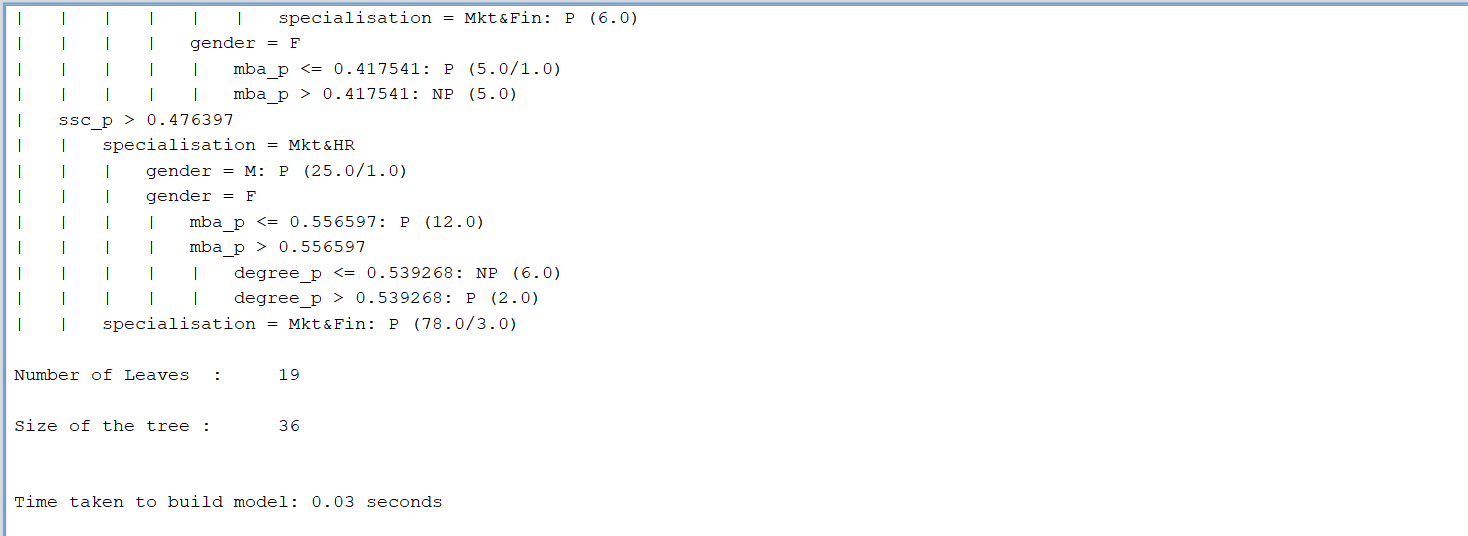
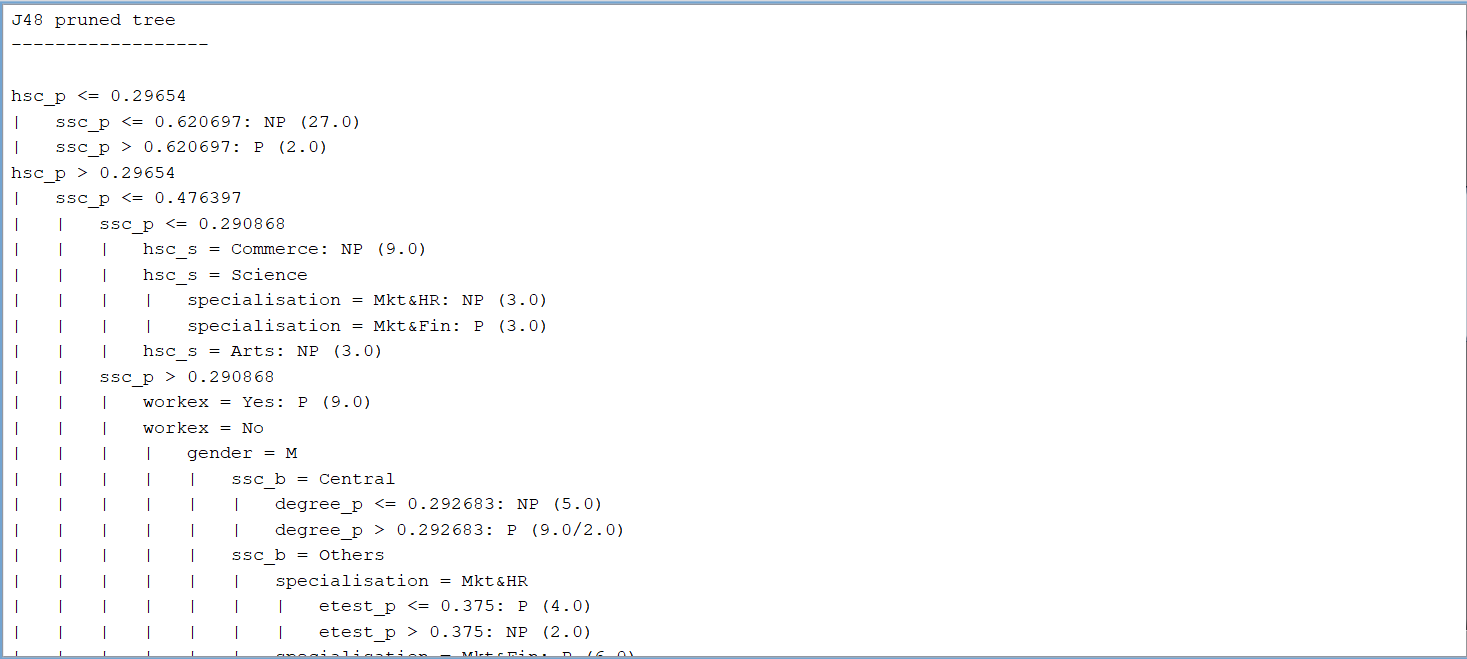
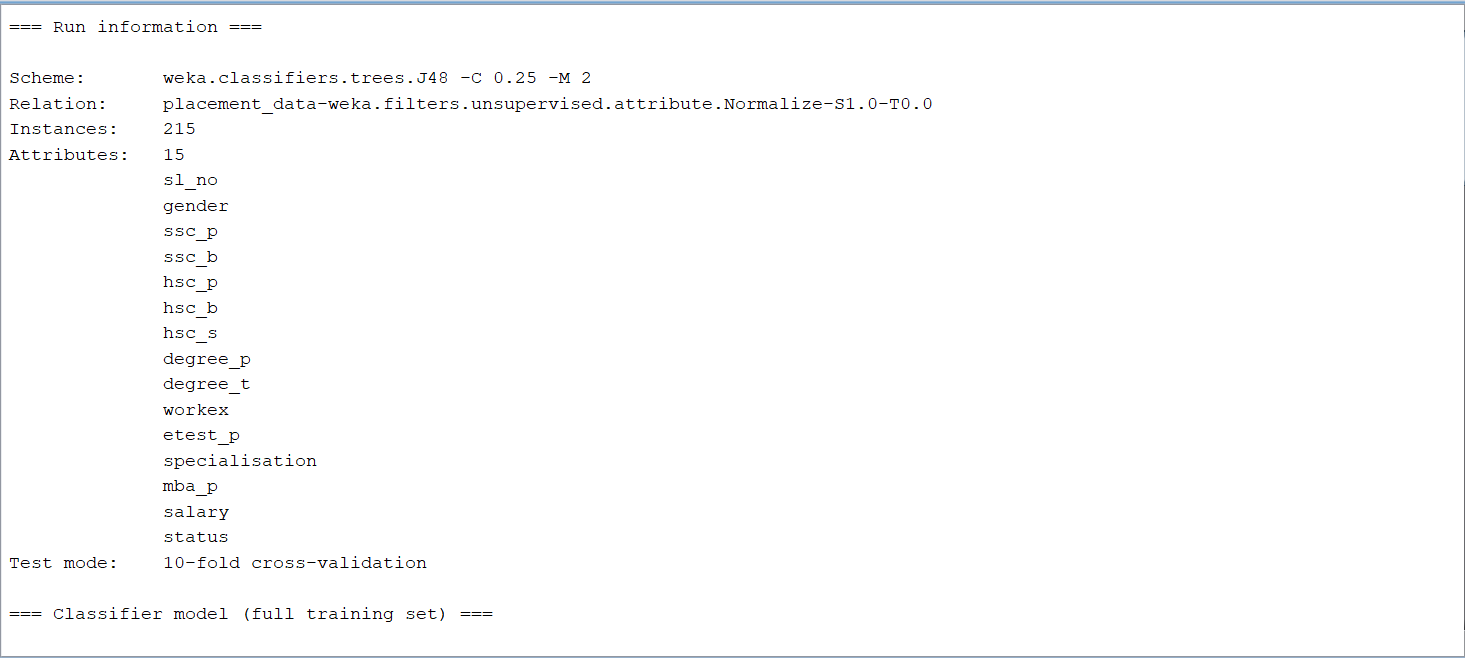
Output:

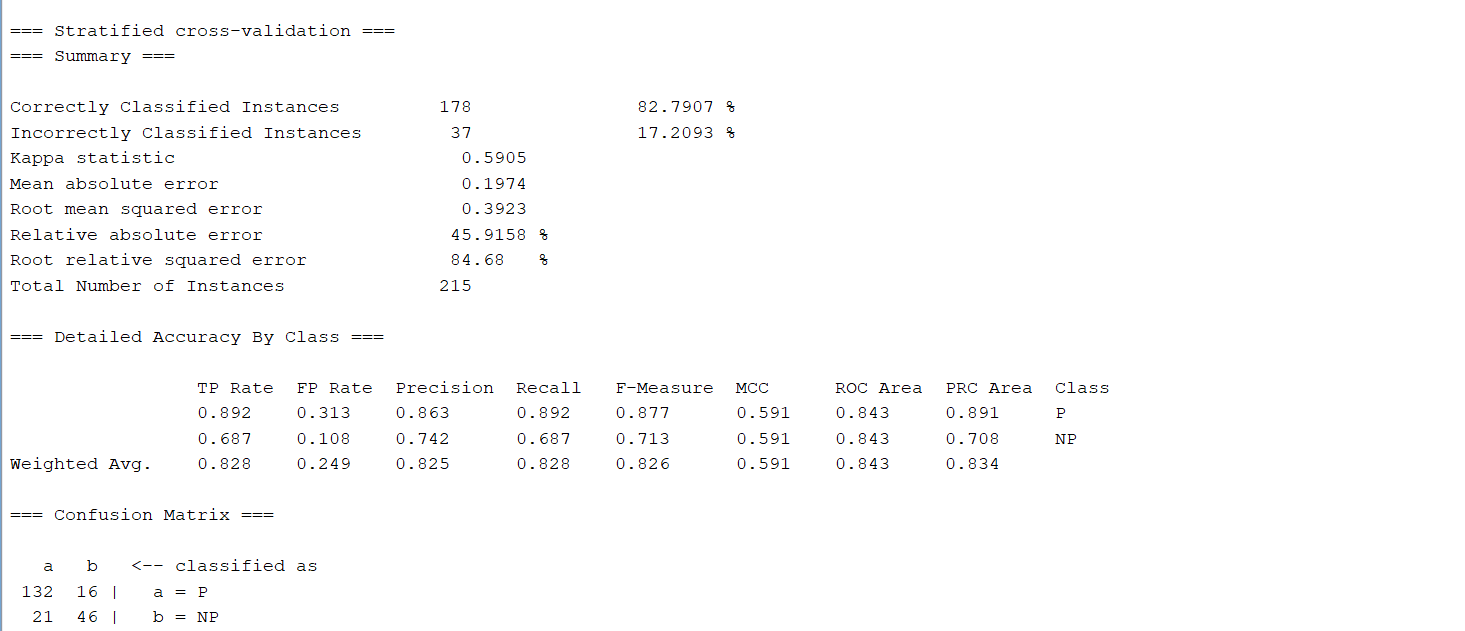


1. Using WEKA tool: For the placement data set given (Placement\_Data.csv), construct a decision tree using J48 and classify the tuple,

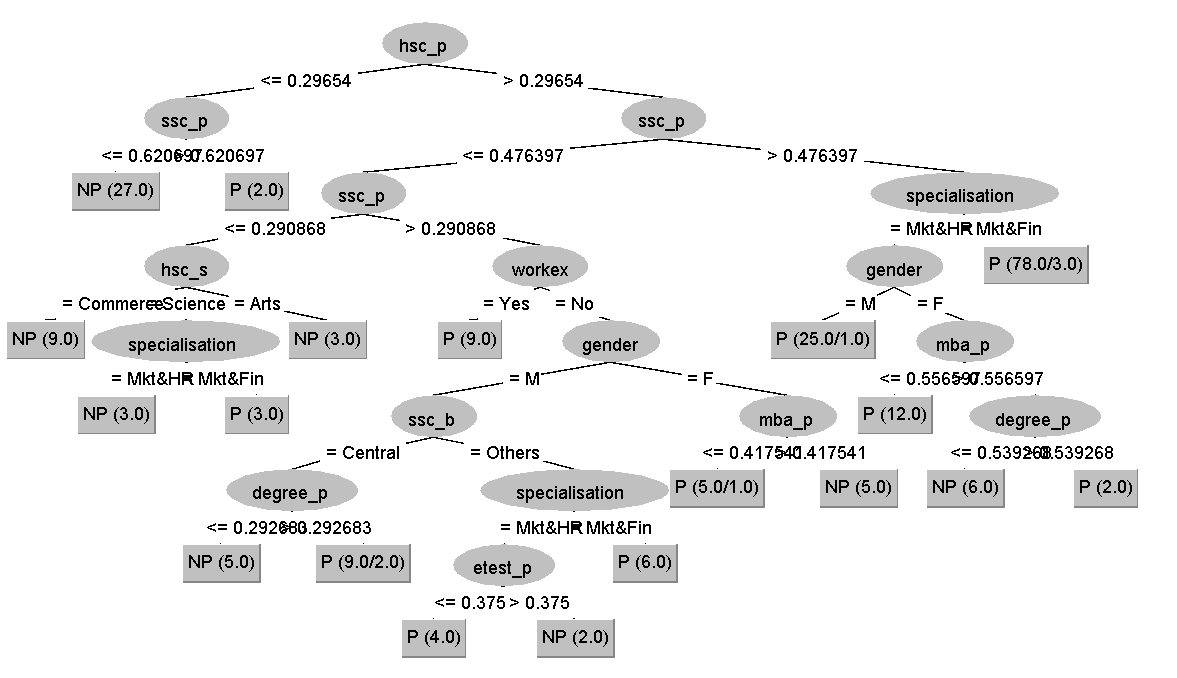
<F,0.950526,Others,0.461285,Others,Science,0.756098,Comm&Mgmt,Yes,0.791667,Mkt&Fin,0.808471,0.081081,Placed>

Evaluation of dataset after Normalizing:



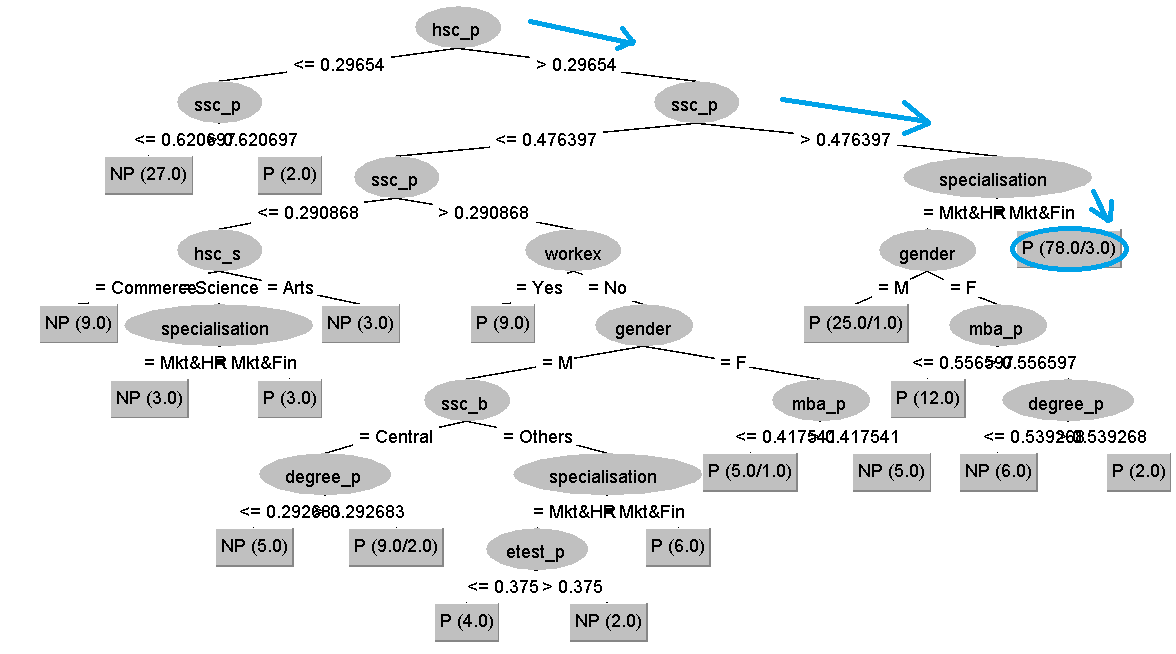


Decision Tree:



**Tuple:** <F,0.950526,Others,0.461285,Others,Science,0.756098,Comm&Mgmt,Yes,0.791667,Mkt&Fin,0.808471,0.081081,Placed>

Solution:



Hence by traversing the tree we classify the tuple as: P (78.0/3.0). That is the tuple is correctly classified as Placed.

**Questions to be answered:**

1. What attributes do you think might be crucial in the decision-making process of classification?

Ans: Nominal and Numeric attributes which provide the most information gain would be the most crucial attributes in the process of classification. In the above question 1 we had a choice between height and gender for the first splitting attribute. We choose the one with the higher info gain, that is height. This lowers the relative error in our tree structure.

Attributes that do not have any info gain or insignificant info gain can be discarded to generate even accurate decision tree.

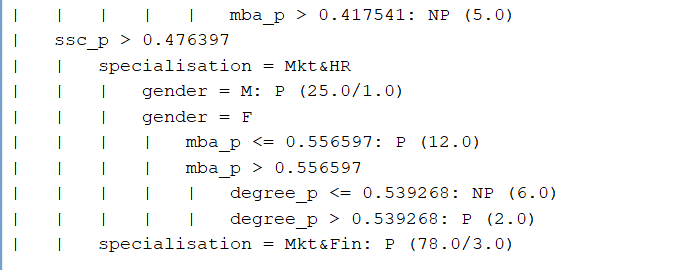
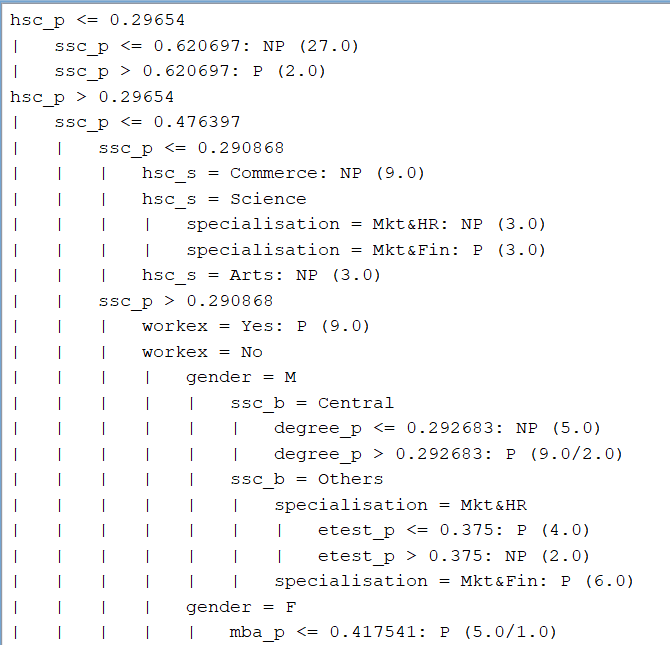
1. Does training a decision tree using cross validation have any improvement on the classification accuracy? Comment.

Ans: Yes, training a decision tree using cross validation can make the decision tree more accurate. Another way could be to discard the attributes with the least information gain.

1. How can you convert the above generated Decision tree into a series of *if - then - rules*

Ans: We can convert a decision tree into a series of if then else statements by traversing through the tree from the top root node down to the leaf nodes.

For the above decision tree, the first if else statements would be: if(hsc\_p<=0.29654) then True(Left ssc\_p) else False(Right ssc\_p). Similarly for ssc\_p: if(ssc\_p<0.620697) then NP(Not Placed) else P(Placed).



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***