Problem 1:
Consider the dataset of printed letters and their features as given at the UCI Machine Learning repository site (http://archive.ics.uci.edu/ml/datasets/Letter+Recognition). This dataset has 20000 instances and 16 attributes. Perform the following tasks and submit the results/answers for each of the following tasks. With each answer state the toolbox / program that you use for getting the answer. Also state the commands, function calls, and parameter values used for obtaining each answer

Solution :
I began with importing the whole data file into a data structure by using

```
ImportedStruct = importdata('lr.data');
```

Then I classified the data into two parts, classes and data matirx by using following command.

```
ClassMat = ImportedStruct.rowheaders;
DataMat =  ImportedStruct.data;
```

1. Normalize the columns for their values to be in uniform ranges. Describe the process you followed to do the normalization.
: To normalize the data from 0-1 series I have used matlab inbuilt function named **normc.**

```
NDataMat = normc(DataMat);
```

2. Split the dataset into three randomly selected parts: 12000 instances for training, 4000 for validation, and 4000 for testing. Describe how you made these partitions.
: I used row by row splitting to get the desired parts. Following is the commands I used.

```
TrDMat = NDataMat(1:12000,:);
TstDMat = NDataMat(12001:16000,:);
ValDMat = NDataMat(16001:20000,:);

TrCMat = ClassMat(1:12000,:);
TstCMat = ClassMat(12001:16000,:);
ValCMat = ClassMat(16001:20000,:);
```

3. Use Matlab (or other toolboxes) to generate decision trees with the following conditions on the leaf nodes of the resulting decision trees: Each leaf node must have at least (all the following cases) 250, 225, 200, 175, 150, 125, 100, 75, 50, 25, 10, and 5 instances of the dataset. Determine the accuracy for each of these decision trees on the training data and the validation data.
: Following is the code I used to generate decision trees with the given conditions, their confusion matrix and to calculate their accuracy

```
%Generation of Decision Trees
MLS250 = fitctree(TrDMat, TrCMat,'MinLeafSize',250);
MLS225 = fitctree(TrDMat, TrCMat,'MinLeafSize',225);
MLS200 = fitctree(TrDMat, TrCMat,'MinLeafSize',200);
MLS175 = fitctree(TrDMat, TrCMat,'MinLeafSize',175);
MLS150 = fitctree(TrDMat, TrCMat,'MinLeafSize',150);
MLS125 = fitctree(TrDMat, TrCMat,'MinLeafSize',125);
MLS100 = fitctree(TrDMat, TrCMat,'MinLeafSize',100);
MLS75 = fitctree(TrDMat, TrCMat,'MinLeafSize',75);
MLS50 = fitctree(TrDMat, TrCMat,'MinLeafSize',50);
MLS25 = fitctree(TrDMat, TrCMat,'MinLeafSize',25);
MLS10 = fitctree(TrDMat, TrCMat,'MinLeafSize',10);
MLS5 = fitctree(TrDMat, TrCMat,'MinLeafSize',5);
```

```
%Accuracy = (1-error)*100
AccuracyMLS5 = (1-resubLoss(MLS5))*100;
AccuracyMLS10 = (1-resubLoss(MLS10))*100;
AccuracyMLS25 = (1-resubLoss(MLS25))*100;
AccuracyMLS50 = (1-resubLoss(MLS50))*100;
AccuracyMLS75 = (1-resubLoss(MLS75))*100;
AccuracyMLS100 = (1-resubLoss(MLS100))*100;
AccuracyMLS125 = (1-resubLoss(MLS125))*100;
AccuracyMLS150 = (1-resubLoss(MLS150))*100;
AccuracyMLS175 = (1-resubLoss(MLS175))*100;
AccuracyMLS200 = (1-resubLoss(MLS200))*100;
AccuracyMLS225 = (1-resubLoss(MLS225))*100;
AccuracyMLS250 = (1-resubLoss(MLS250))*100;

%Training Data Confusion Matrix and accuracy
PredMLS5Mat = predict(MLS5,TstDMat);
PredMLS10Mat = predict(MLS10,TstDMat);
PredMLS25Mat = predict(MLS25,TstDMat);
PredMLS50Mat = predict(MLS50,TstDMat);
PredMLS75Mat = predict(MLS75,TstDMat);
PredMLS100Mat = predict(MLS100,TstDMat);
PredMLS125Mat = predict(MLS125,TstDMat);
PredMLS150Mat = predict(MLS150,TstDMat);
PredMLS175Mat = predict(MLS175,TstDMat);
PredMLS200Mat = predict(MLS200,TstDMat);
PredMLS225Mat = predict(MLS225,TstDMat);
PredMLS250Mat = predict(MLS250,TstDMat);

 [CMTD5,classorder] = confusionmat(TstCMat,PredMLS5Mat);
 CMTD10 = confusionmat(TstCMat,PredMLS10Mat);
 CMTD25 = confusionmat(TstCMat,PredMLS25Mat);
 CMTD50 = confusionmat(TstCMat,PredMLS50Mat);
 CMTD75 = confusionmat(TstCMat,PredMLS75Mat);
 CMTD100 = confusionmat(TstCMat,PredMLS100Mat);
 CMTD125 = confusionmat(TstCMat,PredMLS125Mat);
 CMTD150 = confusionmat(TstCMat,PredMLS150Mat);
 CMTD175 = confusionmat(TstCMat,PredMLS175Mat);
 CMTD200 = confusionmat(TstCMat,PredMLS200Mat);
 CMTD225 = confusionmat(TstCMat,PredMLS225Mat);
 CMTD250 = confusionmat(TstCMat,PredMLS250Mat);

 TAcc5= 100*sum(diag(CMTD5))./sum(CMTD5(:));
 TAcc10= 100*sum(diag(CMTD10))./sum(CMTD10(:));
 TAcc25= 100*sum(diag(CMTD25))./sum(CMTD25(:));
 TAcc50= 100*sum(diag(CMTD50))./sum(CMTD50(:));
 TAcc75= 100*sum(diag(CMTD75))./sum(CMTD75(:));
 TAcc100= 100*sum(diag(CMTD100))./sum(CMTD100(:));
 TAcc125= 100*sum(diag(CMTD125))./sum(CMTD125(:));
 TAcc150= 100*sum(diag(CMTD150))./sum(CMTD150(:));
 TAcc175= 100*sum(diag(CMTD175))./sum(CMTD175(:));
 TAcc200= 100*sum(diag(CMTD200))./sum(CMTD200(:));
 TAcc225= 100*sum(diag(CMTD225))./sum(CMTD225(:));
 TAcc250= 100*sum(diag(CMTD250))./sum(CMTD250(:));



 %Validation Data Confusion Matrix and accuracy
PredMLS5VMat = predict(MLS5,ValDMat);
PredMLS10VMat = predict(MLS10,ValDMat);
PredMLS25VMat = predict(MLS25,ValDMat);
PredMLS50VMat = predict(MLS50,ValDMat);
PredMLS75VMat = predict(MLS75,ValDMat);
PredMLS100VMat = predict(MLS100,ValDMat);
PredMLS125VMat = predict(MLS125,ValDMat);
PredMLS150VMat = predict(MLS150,ValDMat);
PredMLS175VMat = predict(MLS175,ValDMat);
```

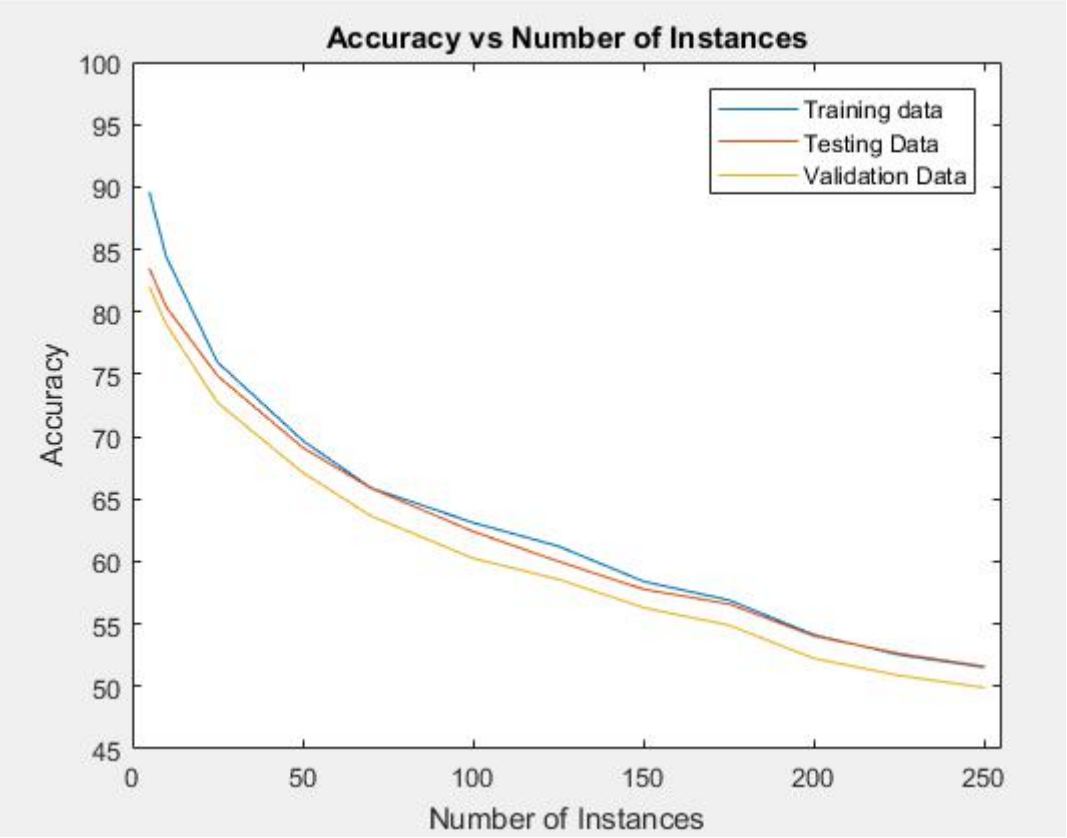
```
PredMLS200VMat = predict(MLS200,ValDMat);
PredMLS225VMat = predict(MLS225,ValDMat);
PredMLS250VMat = predict(MLS250,ValDMat);

 CMVD5 = confusionmat(ValCMat,PredMLS5VMat);
 CMVD10 = confusionmat(ValCMat,PredMLS10VMat);
 CMVD25 = confusionmat(ValCMat,PredMLS25VMat);
 CMVD50 = confusionmat(ValCMat,PredMLS50VMat);
 CMVD75 = confusionmat(ValCMat,PredMLS75VMat);
 CMVD100 = confusionmat(ValCMat,PredMLS100VMat);
 CMVD125 = confusionmat(ValCMat,PredMLS125VMat);
 CMVD150 = confusionmat(ValCMat,PredMLS150VMat);
 CMVD175 = confusionmat(ValCMat,PredMLS175VMat);
 CMVD200 = confusionmat(ValCMat,PredMLS200VMat);
 CMVD225 = confusionmat(ValCMat,PredMLS225VMat);
 CMVD250 = confusionmat(ValCMat,PredMLS250VMat);

 VAcc5= 100*sum(diag(CMVD5))./sum(CMVD5(:));
 VAcc10= 100*sum(diag(CMVD10))./sum(CMVD10(:));
 VAcc25= 100*sum(diag(CMVD25))./sum(CMVD25(:));
 VAcc50= 100*sum(diag(CMVD50))./sum(CMVD50(:));
 VAcc75= 100*sum(diag(CMVD75))./sum(CMVD75(:));
 VAcc100= 100*sum(diag(CMVD100))./sum(CMVD100(:));
 VAcc125= 100*sum(diag(CMVD125))./sum(CMVD125(:));
 VAcc150= 100*sum(diag(CMVD150))./sum(CMVD150(:));
 VAcc175= 100*sum(diag(CMVD175))./sum(CMVD175(:));
 VAcc200= 100*sum(diag(CMVD200))./sum(CMVD200(:));
 VAcc225= 100*sum(diag(CMVD225))./sum(CMVD225(:));
 VAcc250= 100*sum(diag(CMVD250))./sum(CMVD250(:));
```

a. Plot the accuracy values of the trees for all the above cases, and for the training and the validation datasets. How do you interpret the plots?
: Following code is used to plot the graph.

```
AccuracyMat=[AccuracyMLS5,AccuracyMLS10,AccuracyMLS25,AccuracyMLS50,AccuracyMLS75,Accura
cyMLS100,AccuracyMLS125,AccuracyMLS150,AccuracyMLS175,AccuracyMLS200,AccuracyMLS225,Accu
racyMLS250];

AccValData=[VAcc5,VAcc10,VAcc25,VAcc50,VAcc75,VAcc100,VAcc125,VAcc150,VAcc175,VAcc200,VA
cc225,VAcc250];

AccTestData=[TAcc5,TAcc10,TAcc25,TAcc50,TAcc75,TAcc100,TAcc125,TAcc150,TAcc175,TAcc200,T
Acc225,TAcc250];

LeafNodesMat = [5,10,25,50,70,100,125,150,175,200,225,250];

plot(LeafNodesMat,AccuracyMat,'-');

xlim([0 255]);
ylim([45 100]);

xlabel('Number of Instances');
ylabel('Accuracy');

title('Accuracy vs Number of Instances')

hold on;

plot(LeafNodesMat,AccTestData,'-');
hold on;

plot(LeafNodesMat,AccValData,'-');
hold off;
```

```
legend('Training data', 'Testing Data','Validation Data');
```

And following is the graph I achieved.



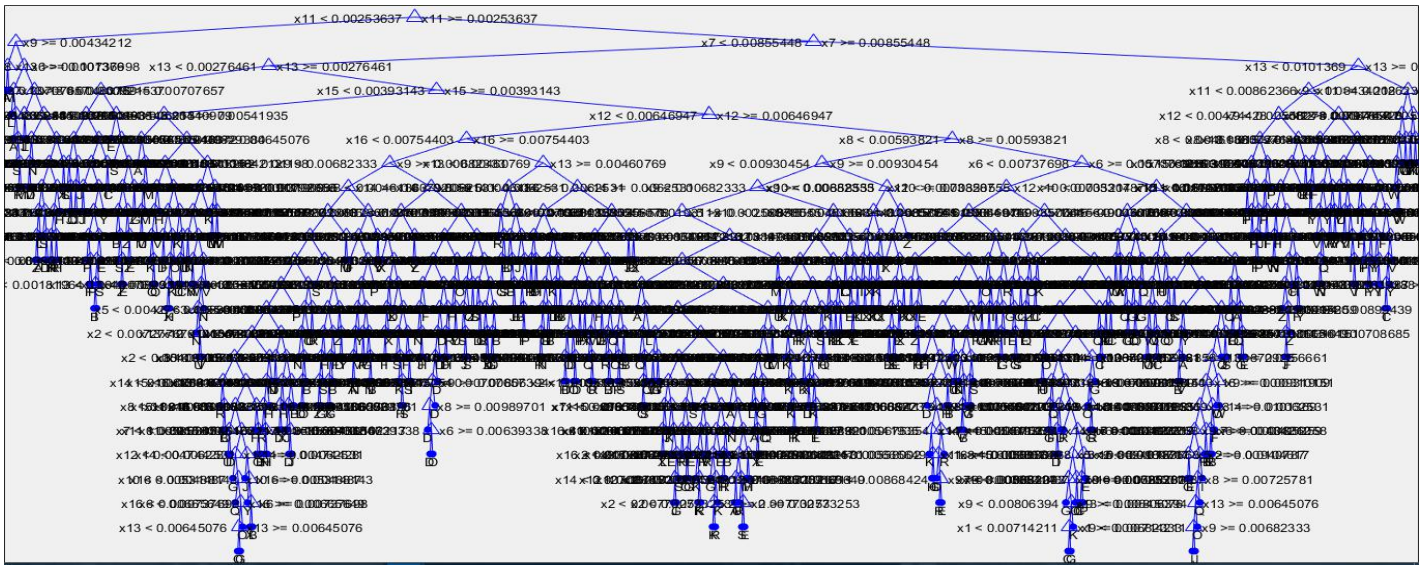
From the graph we can see that the lower the number of instances higher the accuracy. Which is remaining true for test data as well as validation data.

b. Which decision tree is the best from among all the above decision trees and why?

: As stated in the point a, lower the number of instances, higher the accuracy, so for the above example we will select the tree with 5 instances.

c. Show the selected (best) decision tree in tabular (list of rules) or tree structure.

:



d. Use the selected decision tree to compute the Confusion Matrix for the test data set.

: Following is the confusion matrix for the selected tree, the one with 5 instances.

137	0	0	0	1	0	1	3	0	0	0	0	0	3	0	0	3	2	2	1	0	7	0	0	1	2
0	116	0	0	0	0	0	0	0	4	0	1	0	0	2	0	2	0	4	3	0	0	0	0	0	0
1	1	161	0	1	0	0	1	0	0	1	0	2	0	2	1	0	0	2	0	0	1	0	0	0	0
1	0	0	126	0	3	2	0	3	2	0	3	2	0	3	0	1	0	2	0	0	0	1	0	0	3
0	0	1	1	137	3	0	0	0	0	0	1	0	1	0	1	2	0	2	2	0	0	4	0	0	0
3	0	0	0	0	132	0	0	2	0	0	2	0	3	1	0	1	1	1	0	0	1	0	2	0	2
0	1	1	6	0	0	132	0	4	0	0	6	3	0	1	0	0	1	1	0	0	1	2	5	0	6
0	0	0	0	0	1	0	122	0	1	0	0	1	0	1	3	1	0	1	1	0	2	5	1	1	8
1	0	2	2	0	0	1	4	128	1	4	3	1	0	1	0	1	1	3	0	2	0	1	1	0	1
0	3	0	0	0	0	0	2	1	129	0	0	4	0	4	4	2	1	11	0	0	0	0	0	1	1
0	0	0	0	0	0	2	0	4	4	131	3	4	0	2	1	1	0	2	0	0	0	0	0	0	0
0	2	0	3	2	0	5	0	0	0	0	118	2	0	5	0	0	1	2	4	0	0	1	3	0	3
0	0	0	1	0	0	2	5	3	4	0	0	125	0	0	1	3	0	1	0	2	2	0	0	0	1
1	0	0	0	4	3	0	0	1	0	0	0	0	151	1	0	1	1	0	1	0	0	0	3	0	0
2	1	0	3	0	0	2	1	1	3	2	7	0	0	127	1	0	4	5	0	0	0	3	0	1	2
0	1	1	0	6	0	0	0	1	2	0	1	1	0	0	133	0	1	1	0	6	1	1	0	2	0
2	2	0	2	0	0	2	0	0	0	2	0	4	0	0	0	141	8	0	0	2	5	0	0	0	0
3	1	0	1	0	0	2	0	0	0	0	1	0	0	0	0	9	125	2	0	0	0	0	1	0	1
0	7	0	3	0	0	0	1	1	4	0	0	0	1	2	2	4	2	127	0	1	1	2	0	0	0
0	1	1	1	5	2	1	1	0	0	0	1	0	0	1	0	0	0	1	134	1	0	0	8	0	1
0	0	1	0	0	0	1	1	2	0	1	1	1	0	0	3	0	1	0	1	121	0	0	0	2	2
4	0	0	0	0	2	1	1	2	0	1	0	3	0	0	0	2	2	0	2	0	125	0	0	1	0
0	1	1	8	1	2	3	3	0	0	0	1	0	0	0	1	2	1	3	0	0	0	117	0	0	4
2	0	0	0	3	2	2	0	0	0	0	1	0	3	1	0	0	0	0	1	0	0	0	131	0	3
0	0	0	1	0	0	1	0	3	1	2	0	3	0	1	0	6	0	2	0	10	2	1	0	124	2
0	0	0	3	0	0	6	3	0	0	0	7	1	0	1	0	0	1	1	1	0	1	1	0	0	89

e. Compute Precision, Recall, and F metrics for any randomly selected three of the 26 classes.

: Following is the code I have used to calculate precision, Recall and F1 Score

```
%Precision Calculation
Precision = diag(CMTD5)./sum(CMTD5,2); %for order 'classorder'

%Recall Calculation
Recall = diag(CMTD5)./sum(CMTD5,1); %for order 'classorder'

%F1-Score Calculation
F1score = 2*(Precision.*Recall)./(Precision+Recall);%for order 'classorder'
```

Following are the precision, recall and F1 score values for 3 random classes.

Class Name	Precision	Recall	F1 Score
P	0.8562	0.8394	0.8364
Q	0.7697	0.8232	0.8283
R	0.7905	0.9513	0.8885

Problem 2:

Consider the Pima Indians Diabetes dataset from the UCI Machine Learning repository. (https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes) This dataset has 768 Instances and 8 attributes. Find the decision tree that has at least 15 instances at each leaf node by doing a 10-fold cross-validation while learning the decision tree.

Solution :

I began with importing the whole data file into a data structure by using

```
ImportedData = importdata('pima-indians-diabetes.data');%ImportData
```

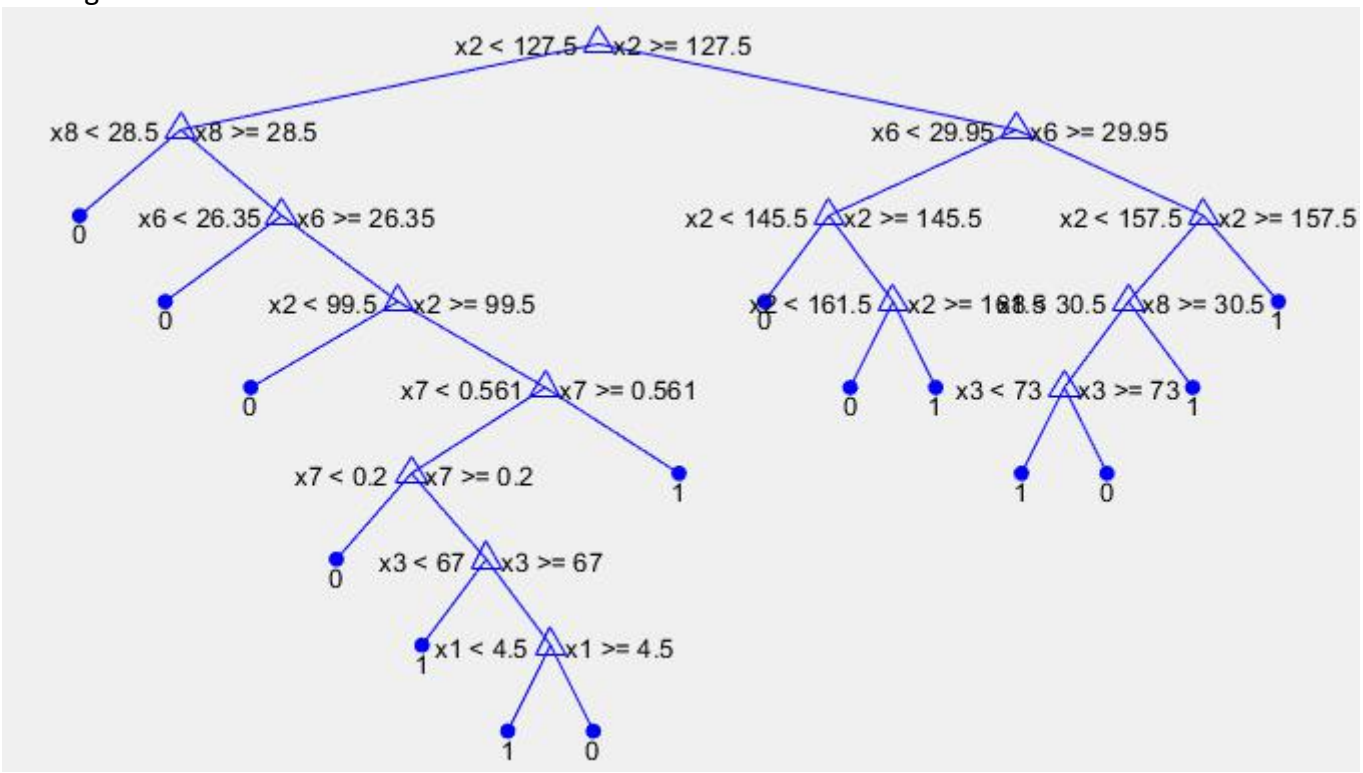
Then I classified the data into two parts, classes and data matirx by using following command.

```
DataMat = ImportedData(:,1:8);
ClassMat = ImportedData(:,9);
```

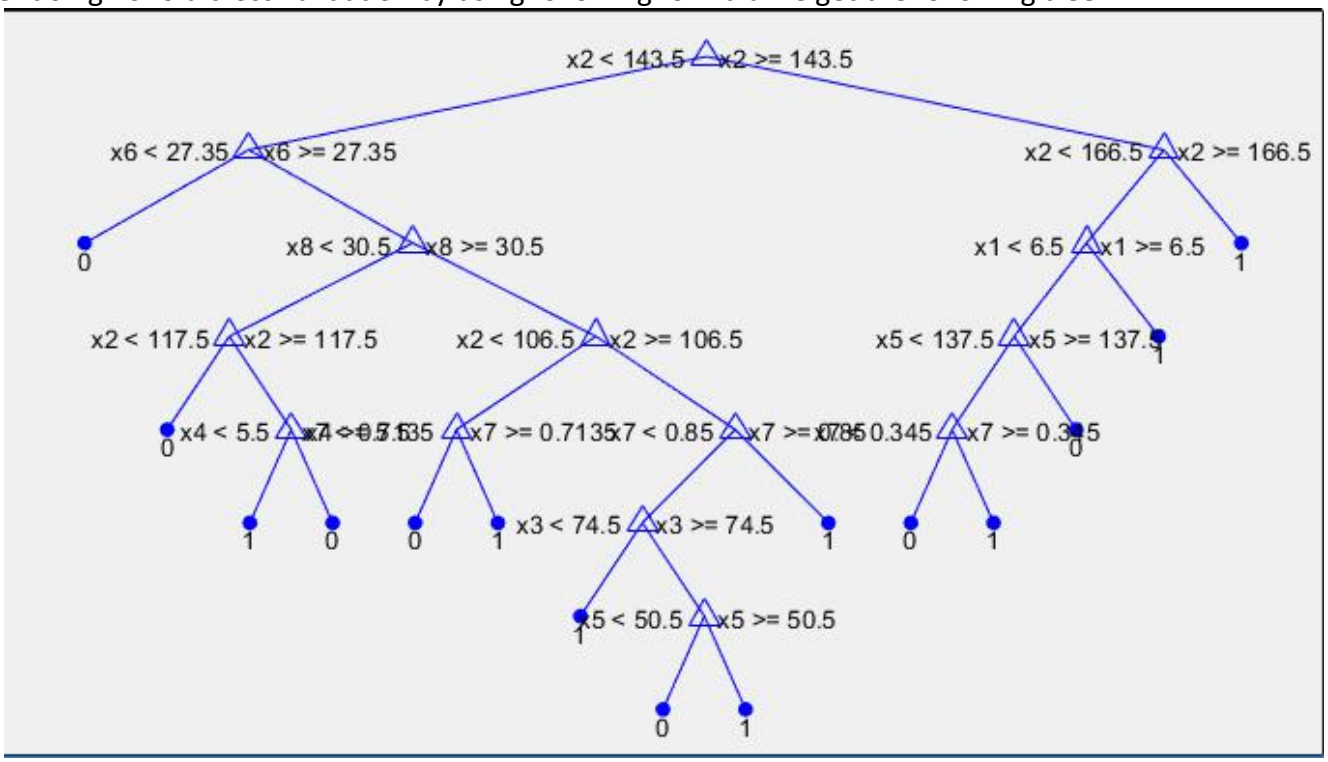
a. Show the decision tree in the table/list/tree form.
: Following is the code I have used to generate a decision tree with at least 15 instances and to present it.

```
MLS15 = fitctree(DataMat, ClassMat,'MinLeafSize',15);
view(MLS15,'Mode','graph');
```

Following is the tree I achieved from the above code.



After doing 10 fold cross validation by using following formula we get the following tree



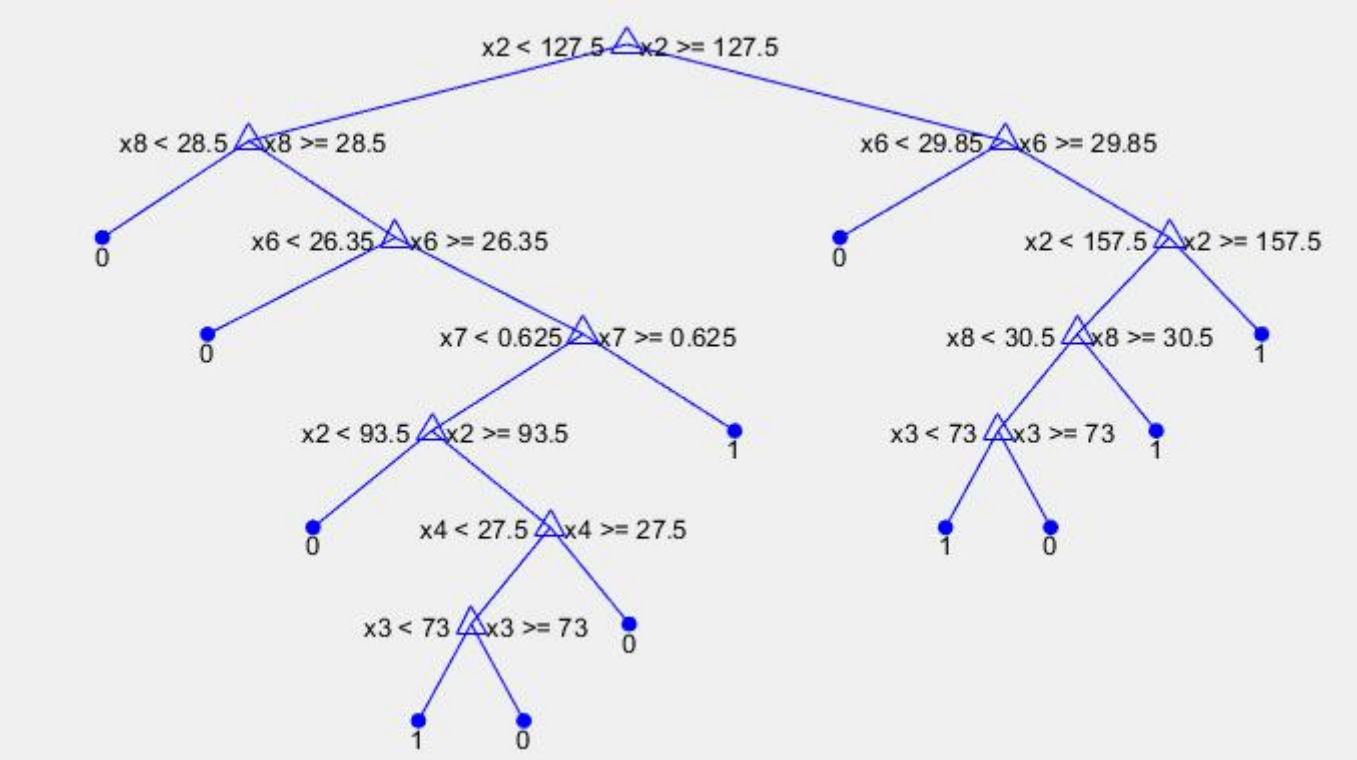
b. Find the best possible decision tree by adopting the appropriate parameter values. Show this decision tree. Show the parameters that yield this tree and also list its precision, recall, and accuracy values.
: For the above tree I calculated accuracy by calculating KFoldloss by using following formula which comes to the value of 75.7813%.

```
AccuracyCVMLS15 = (1-(kfoldLoss(CVMLS15)))*100;
```

We can increase the accuracy of this decision tree by varying the leafsize and k-fold cross validation
New Leaf Size : 20
New K-Fold Value : 0

Following is the code I used and achieved the accuracy of 78.56%.

```
BDTree = fitctree(DataMat, ClassMat,'MinLeafSize',20,'KFold',8);
view(BDTree.Trained{1},'Mode','graph');
AccuracyBDTree = (1-(kfoldLoss(BDTree)))*100;
```



Following is the code I have used to calculate precision, Recall and F1 Score.
First I classified the datamatrix into 2 parts, training and sample. Then generated a confusion matrix and calculated the precision, recall and f1 score.

Following are the precision, recall and F1 score values for the class 0 and 1.

Class Name	Precision	Recall	F1 Score
0	0.8123	0.6209	0.6678
1	0.7398	1.0821	0.8747