Assignment 1

Harshith Bondada
M10724854
Environment: Matlab

1. (30) Take Data2 and split it into randomly selected 210 training instances and remaining 100 as test instance. Create decision trees using the training set and the “minimum records per leaf node” values of 5, 10, 15, 20, and 25.

Answer:
I have imported the data using the Import Button.
For space reasons, I am showing only a partial amount of the dataset.

210×7 table

pelvic_incidence	pelvic_tiltnumeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
72.054	24.701	79.874	47.353	107.17	56.426
72.644	18.929	68	53.715	116.96	25.384
61.735	17.114	46.9	44.621	120.92	3.0877
44.431	14.174	32.243	30.256	131.72	-3.6043
54.504	6.8199	47	47.684	111.79	-4.4068
61.412	25.384	39.097	36.027	103.4	21.843
54.742	12.095	41	42.647	117.64	40.382
50.21	29.76	36.104	20.45	128.29	5.7406
67.028	13.282	66.15	53.746	100.72	33.989
43.436	10.096	36.032	33.341	137.44	-3.1145
63.835	20.363	54.552	43.472	112.31	-0.62253
67.538	14.655	58.001	52.883	123.63	25.97
37.14	16.481	24	20.659	125.01	7.3664
51.325	13.631	33.259	37.694	131.31	1.7889
82.407	29.276	77.055	53.13	117.04	62.765
40.747	1.8355	50	38.911	139.25	0.66856
69.781	13.777	58	56.004	118.93	17.915
52.419	19.012	35.873	33.408	116.56	1.6947
43.192	9.9767	28.938	33.215	123.47	1.741
78.492	22.182	60	56.31	118.53	27.383
43.718	9.812	52	33.906	88.434	40.881
44.216	1.5071	46.11	42.709	108.63	42.81
69.399	18.898	75.966	50.5	103.58	-0.44366
58.102	14.838	79.65	43.264	113.59	50.238
65.536	24.157	45.775	41.379	136.44	16.378
84.974	33.021	60.86	51.953	125.66	74.333
42.918	-5.846	58	48.764	121.61	-3.362
57.146	16.489	42.842	40.657	113.81	5.0152
50.677	6.4615	35	44.215	116.59	-0.21471
58.6	-0.2615	51.5	58.861	102.04	28.06
39.657	16.209	36.675	23.448	131.92	-4.969
81.082	21.256	78.767	59.826	90.072	49.159

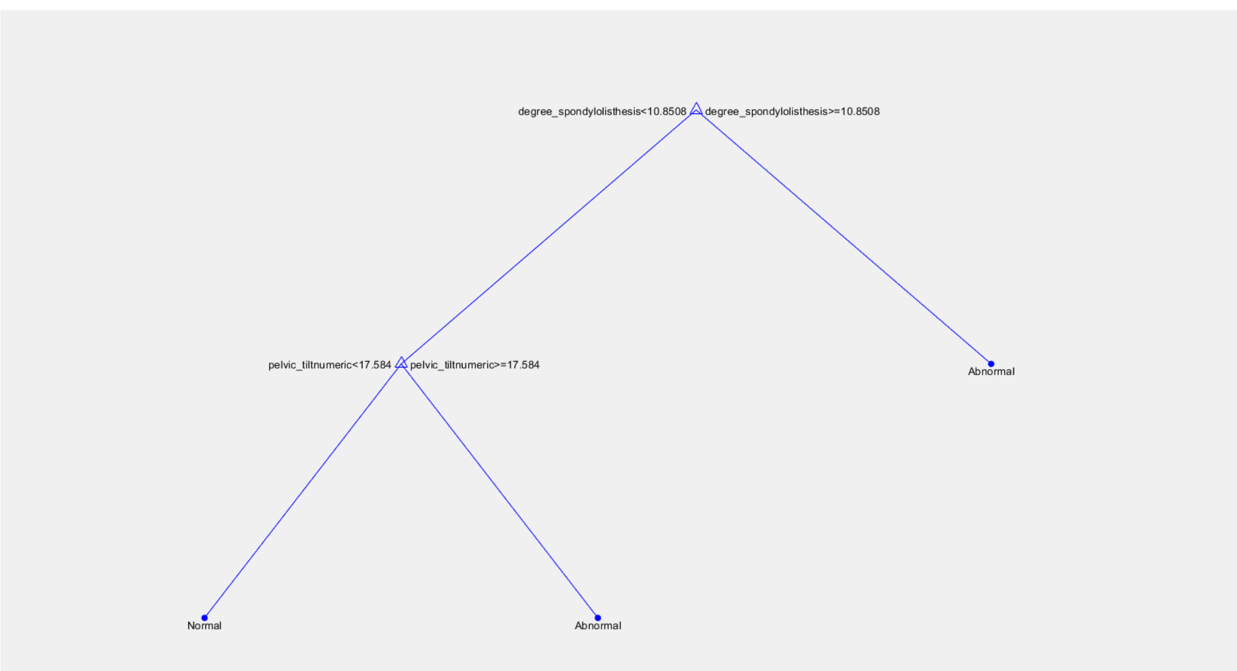
class
Abnormal
Abnormal
Normal
Normal
Normal
Abnormal
Abnormal
Abnormal
Abnormal
Normal
Abnormal
Normal
Normal
Normal
Abnormal
Normal
Abnormal
Abnormal
Normal
Abnormal
Abnormal
Abnormal
Normal
Abnormal
Abnormal
Abnormal
Normal
Normal
Normal
Abnormal
Normal
Abnormal
Normal

The below screenshot is the Matlab code for creation of the decision trees with 5, 10,15,20,25 leaf node; I have used the function fitctree to generate the decision tree using the arguments such as MinLeafSize as 5, 10,15,20,25 to limit leaf nodes.

```
%Decision Trees Constructing
load Data2_training
DTC_5=fitctree(Data2_training, 'class','MinLeafSize',3);
DTC_10=fitctree(Data2_training, 'class','MinLeafSize',10);
DTC_15=fitctree(Data2_training, 'class','MinLeafSize',15);
DTC_20=fitctree(Data2_training, 'class','MinLeafSize',20);
DTC_25=fitctree(Data2_training, 'class','MinLeafSize',25);
```

a. Show the tree for the value 25. Comment on what you notice about the five trees.

The below figure is the screenshot for Tree value with node 25.
I have noticed that the as long the Min leaf node increases the tree height decreases and the child nodes too.



b. For each tree compute and report the accuracy, precision, and recall values. Comment on the comparison of these values and show these values on a plot.

I have computed the confusion matrix and calculated the accuracy, precision, recall values.

Accuracy:

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.
Accuracy = TP+TN/TP+FP+FN+TN

Precision:

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.
Precision = TP/TP+FP

Recall:
Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

Recall = TP/TP+FN

I have computed the confusion matrices using the “confusionmat” function and calculated the Accuracy, Recall, and Precision using the formula as shown in the screenshot.

```
    %Confusion Matrics
    Actual_D = Data2_training{1:210,{'class'}};
    %Confusion Matrix for Decision Tree with 5 nodes
    prediction_5=predict(DTC_5,Data2_training);
    C5=confusionmat(Actual_D, prediction_5);

    %Confusion Matrix for Decision Tree with 10 nodes
    prediction_10=predict(DTC_10,Data2_training);
    C10=confusionmat(Actual_D, prediction_10);

    %Confusion Matrix for Decision Tree with 15 nodes
    prediction_15=predict(DTC_15,Data2_training);
    C15=confusionmat(Actual_D, prediction_15);

    %Confusion Matrix for Decision Tree with 20 nodes
    prediction_20=predict(DTC_20,Data2_training);
    C20=confusionmat(Actual_D, prediction_20);

    %Confusion Matrix for Decision Tree with 25 nodes
    prediction_25=predict(DTC_25,Data2_training);
    C25=confusionmat(Actual_D, prediction_25);

%Calculating recall performances

%Calculating recall performances for Decision tree with 5 nodes
recall_DecisionTree_5=C5(2,2)/(C5(2,2)+C5(2,1));

%Calculating recall performances for Decision tree with 10 nodes
recall_DecisionTree_10=C10(2,2)/(C10(2,2)+C10(2,1));

%Calculating recall performances for Decision tree with 15 nodes
recall_DecisionTree_15=C15(2,2)/(C15(2,2)+C15(2,1));

%Calculating recall performances for Decision tree with 20 nodes
recall_DecisionTree_20=C20(2,2)/(C20(2,2)+C20(2,1));

%Calculating recall performances for Decision tree with 25 nodes
recall_DecisionTree_25=C25(2,2)/(C25(2,2)+C25(2,1));
```

```
%Calculating accuracy performances

%Calculating accuracy performances for Decision tree with 5 nodes
accuracy_DecisionTree_5=(C5(1,1)+C5(2,2))/(C5(1,1)+C5(1,2)+C5(2,1)+C5(2,2));
%Calculating accuracy performances for Decision tree with 10 nodes
accuracy_DecisionTree_10=(C10(1,1)+C10(2,2))/(C10(1,1)+C10(1,2)+C10(2,1)+C10(2,2));
%Calculating accuracy performances for Decision tree with 15 nodes
accuracy_DecisionTree_15=(C15(1,1)+C15(2,2))/(C15(1,1)+C15(1,2)+C15(2,1)+C15(2,2));
%Calculating accuracy performances for Decision tree with 20 nodes
accuracy_DecisionTree_20=(C20(1,1)+C20(2,2))/(C20(1,1)+C20(1,2)+C20(2,1)+C20(2,2));
%Calculating accuracy performances for Decision tree with 25 nodes
accuracy_DecisionTree_25=(C25(1,1)+C25(2,2))/(C25(1,1)+C25(1,2)+C25(2,1)+C25(2,2));

%Calculating percision performances

%Calculating percision performances for Decision tree with 5 nodes
precision_D5=C5(2,2)/(C5(2,2)+C5(1,2));
%Calculating percision  performances for Decision tree with 10 nodes
precision_D10=C10(2,2)/(C10(2,2)+C10(1,2));
%Calculating percision  performances for Decision tree with 15 nodes
precision_D15=C15(2,2)/(C15(2,2)+C15(1,2));
%Calculating percision  performances for Decision tree with 20 node
precision_D20=C20(2,2)/(C20(2,2)+C20(1,2));
%Calculating percision  performances for Decision tree with 25 nodes
precision_D25=C25(2,2)/(C25(2,2)+C25(1,2));
```

The accuracy of the Decision tree with 5 nodes is 0.9333 which is the highest value when we compare to any accuracy value with any other Decision Tree.

The second highest is 0.8714 with both the Decision tree having 10 and 15 leaf nodes.

accuracy_DecisionTree_15	0.8714
accuracy_DecisionTree_20	0.8571
accuracy_DecisionTree_25	0.8381
accuracy_DecisionTree_5	0.9333
accuracy_DecisionTree_10	0.8714

Coming to precision values,
We find that the decision tree with 5 leaf nodes has highest 0.8730. The values were 87% correct when predicted.
The least is decision tree with leaf nodes 25. It has the least value of 70%.

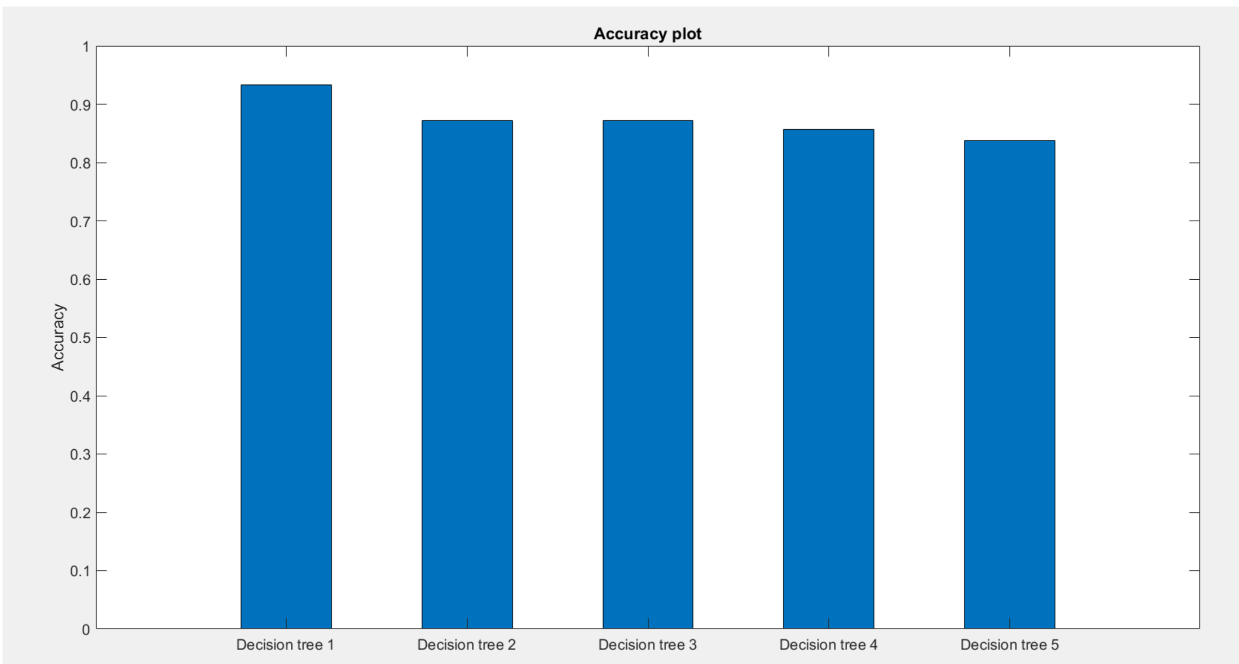
precision_D15	0.8269
precision_D20	0.8444
precision_D25	0.7015
precision_D5	0.8730
precision_D10	0.8269

Recall value is high for the Decision tree with 5 leaf nodes.
Similar, to the precision values decision tree with 25 leaf nodes has the least recall value.

recall_DecisionTree_15	0.7049
recall_DecisionTree_20	0.6230
recall_DecisionTree_25	0.7705
recall_DecisionTree_5	0.9016
recall_DecisionTree_10	0.7049

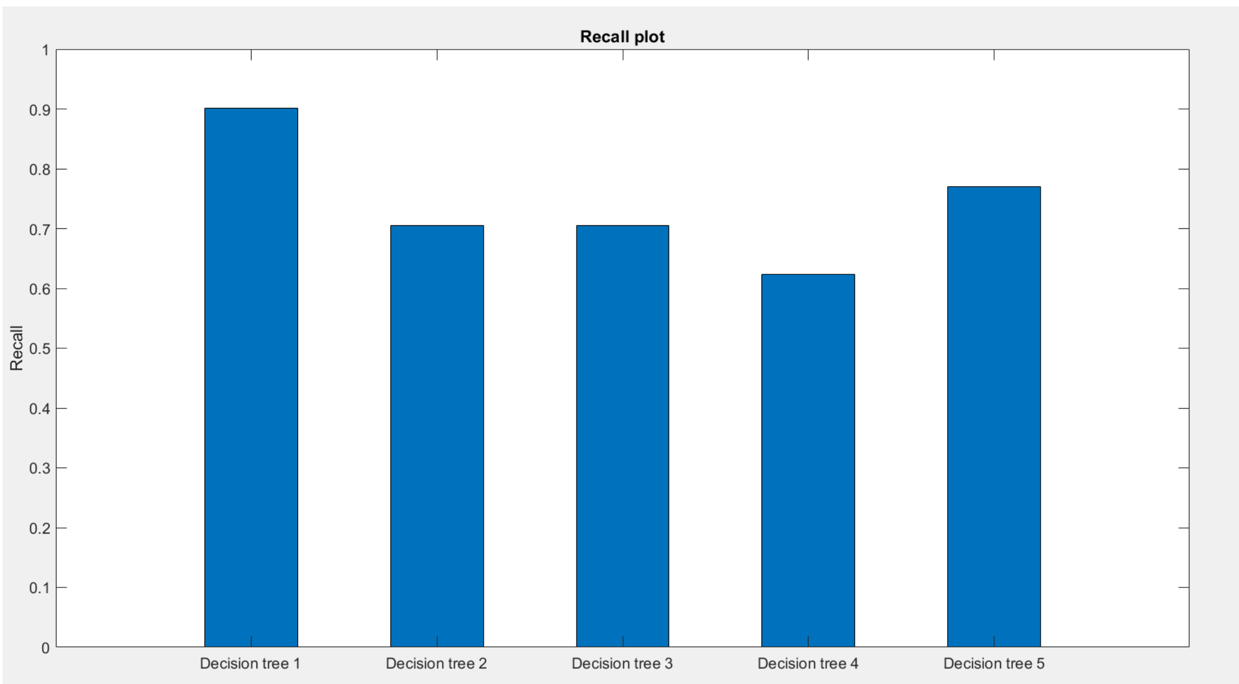
The Accuracy plot of the Decision Tree are shown below through bar graph, x-axis represent the decision trees starting from 5 leaf nodes till 25 leaf nodes.
Y-axis represent the accuracy values.

We can clearly see that the
Decision tree with 5 leaf nodes has the highest accuracy value.

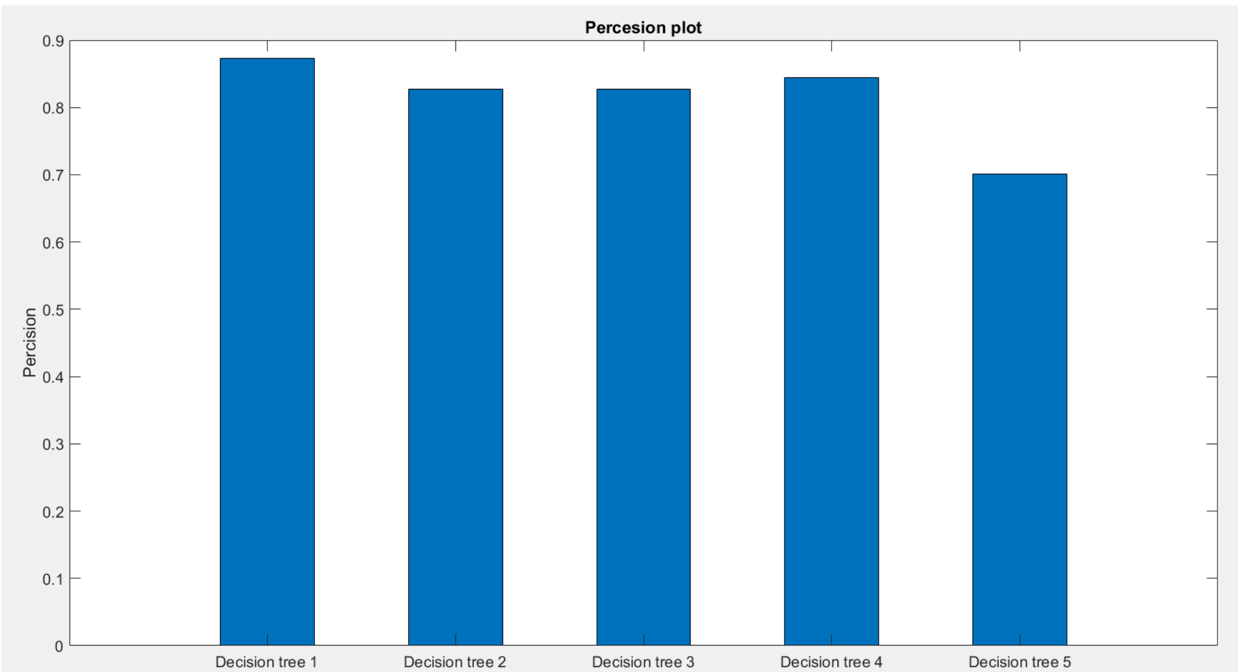


The Recall plot of the Decision Tree are shown below through bar graph, x-axis represent the decision trees starting from 5 leaf nodes till 25 leaf nodes.
Y-axis represent the recall values.

We can see in the plot that the Decision tree 1 has the highest Recall value.



The below plot represents the Precision Plot of the Decision Trees.
The Precision plot of the Decision Tree are shown below through bar graph, x-axis represent the decision trees starting from 5 leaf nodes till 25 leaf nodes.
Y-axis represent the precision values.



c. Now limit yourself to the case of 10 minimum records per leaf node. Repeat the tree learning exercise five times by randomly choosing different sets of 210 training instances. Report the accuracy, precision, and recall values for each run and also their averages and standard deviations. Comment on the variability of the values as the random sample changes.

Answer:
The below is the screen shot code for generating 5 different random training samples. The code snippet also does the following:

- Constructing a decision tree with 10 leaf nodes.
- Calculating confusion matrix.
- Calculating the accuracy, recall, precision values.

```
%Generating 5 datasets

Generating 1st dataset for traing data
k1 = randperm(size(Data2,1));%Creating a random permutation vector
Data2_training_1 = Data2 (k1(1: 210), :);%Dividing the dataset2 into 210 training rows

load Data2_training_1
DTC_10_1=fitctree(Data2_training_1, 'class','MinLeafSize',10);%Decision tree of 10 nodes with 2nd training data

%Confusion Matrics generation with variables with 2nd training data
Actual_D_1 = Data2_training_1{1:210,{'class'}};
prediction_10_1=predict(DTC_10_1,Data2_training_1);

%Confusion Matrix for Decision Tree with 10 nodes
[C10_1,C10_1_order]=confusionmat(Actual_D_1, prediction_10_1);

%Calculating recall performances with seccond training data
recall_DecisionTree_10_1=C10_1(2,2)/(C10_1(2,2)+C10_1(2,1));


%Calculating accuracy performances
accuracy_DecisionTree_10_1=(C10_1(1,1)+C10_1(2,2))/(C10_1(1,1)+C10_1(1,2)+C10_1(2,1)+C10_1(2,2));

%Calculating percision performances
precision_D10_1=C10_1(2,2)/(C10_1(2,2)+C10_1(1,2));
```

The below screen snippet show that the Data2 has been randomized and generated a table with 210 training samples. I have named it as Data2_training_1 since, this is one of the 5 random datasets.

Data2_training_1 210x7 table

A decision tree has been constructed and I have named it as DTC_10_1. 10 means that with 10 leaf nodes. _1 means one of the 5 decision tree.

DTC_10_1 1x1 ClassificationTree

The accuracy, precision, recall parameters of the decision tree are shown below.

accuracy_DecisionTree_10_1 0.9095

precision_D10_1 0.8551

recall_DecisionTree_10_1 0.8676

The below is the screen shot code for generating 2nd random training sample. The code snippet also does the following:

- Constructing a decision tree with 10 leaf nodes.
- Calculating confusion matrix.
- Calculating the accuracy, recall, precision values.

```
Generating 2nd dataset for traing data
k2 = randperm(size(Data2,1));%Creating a random permutation vector
Data2_training_2 = Data2 (k2(1: 210), :);%Dividing the dataset2 into 210 training rows

load Data2_training_2
DTC_10_2=fitctree(Data2_training_2, 'class','MinLeafSize',10);%Decision tree of 10 nodes with 3rd training data

%Confusion Matrics generation with variables with 3rd training data
Actual_D_2 = Data2_training_2{1:210,{'class'}};
prediction_10_2=predict(DTC_10_2,Data2_training_2);

%Confusion Matrix for Decision Tree with 10 nodes
C10_2=confusionmat(Actual_D_2, prediction_10_2);

%Calculating recall performances with seccond training data
recall_DecisionTree_10_2=C10_2(2,2)/(C10_2(2,2)+C10_2(2,1));


%Calculating accuracy performances
accuracy_DecisionTree_10_2=(C10_2(1,1)+C10_2(2,2))/(C10_2(1,1)+C10_2(1,2)+C10_2(2,1)+C10_2(2,2));

%Calculating percision performances

precision_D10_2=C10_2(2,2)/(C10_2(2,2)+C10_2(1,2));
```

For naming conventions, I have named the 2nd randomly generated dataset as Data2_training_2.

Data2_training_2	210x7 table

The respective decision tree has been generated.

DTC_10_2	1x1 ClassificationTree

The Accuracy, Precision, Recall of the 2nd Decision tree has been computed.

accuracy_DecisionTree_10_2	0.8619
precision_D10_1	0.8551
recall_DecisionTree_10_2	0.7945

The below is the screen shot code for generating 3rd random training sample. The code snippet also does the following:

- Constructing a decision tree with 10 leaf nodes.
- Calculating confusion matrix.
- Calculating the accuracy, recall, precision values.

```
Generating 3rd dataset for traing data
k3 = randperm(size(Data2,1));%Creating a random permutation vector
Data2_training_3 = Data2 (k3(1: 210), :);%Dividing the dataset2 into 210 training rows

load Data2_training_3
DTC_10_3=fitctree(Data2_training_3, 'class','MinLeafSize',10);%Decision tree of 10 nodes with 3rd training data

%Confusion Matrics generation with variables with 3rd training data
Actual_D_3 = Data2_training_3{1:210,{'class'}};
prediction_10_3=predict(DTC_10_3,Data2_training_3);

%Confusion Matrix for Decision Tree with 10 nodes
C10_3=confusionmat(Actual_D_3, prediction_10_3);

%Calculating recall performances with seccond training data
recall_DecisionTree_10_3=C10_3(2,2)/(C10_3(2,2)+C10_3(2,1));


%Calculating accuracy performances
accuracy_DecisionTree_10_3=(C10_3(1,1)+C10_3(2,2))/(C10_3(1,1)+C10_3(1,2)+C10_3(2,1)+C10_3(2,2));

%Calculating percision performances

precision_D10_3=C10_3(2,2)/(C10_3(2,2)+C10_3(1,2));
```

For naming conventions, I have named the 3rd randomly generated dataset as Data2_training_2.

Data2_training_3	*210x7 table*

The respective decision tree has been constructed.

DTC_10_3	*1x1 ClassificationTree*

The accuracy, recall, precision are as follows:

accuracy_DecisionTree_10_3	0.8905
precision_D10_3	0.8421
recall_DecisionTree_10_3	0.7742

The below is the screen shot code for generating 4th random training sample. The code snippet also does the following:

- Constructing a decision tree with 10 leaf nodes.
- Calculating confusion matrix.
- Calculating the accuracy, recall, precision values.

```
Generating 4th dataset for traing data
k4 = randperm(size(Data2,1));%Creating a random permutation vector
Data2_training_4 = Data2 (k4(1: 210), :);%Dividing the dataset2 into 210 training rows

load Data2_training_4
DTC_10_4=fitctree(Data2_training_4, 'class','MinLeafSize',10);%Decision tree of 10 nodes with 3rd training data

%Confusion Matrics generation with variables with 3rd training data
Actual_D_4 = Data2_training_4{1:210,{'class'}};
prediction_10_4=predict(DTC_10_4,Data2_training_4);

%Confusion Matrix for Decision Tree with 10 nodes
C10_4=confusionmat(Actual_D_4, prediction_10_4);

%Calculating recall performances with seccond training data
recall_DecisionTree_10_4=C10_4(2,2)/(C10_4(2,2)+C10_4(2,1));


%Calculating accuracy performances
accuracy_DecisionTree_10_4=(C10_4(1,1)+C10_4(2,2))/(C10_4(1,1)+C10_4(1,2)+C10_4(2,1)+C10_4(2,2));

%Calculating percision performances

precision_D10_4=C10_4(2,2)/(C10_4(2,2)+C10_4(1,2));
```

I have named the training data set as the 4th one, since, it is the fourth one.

Data2_training_4 210x7 table

A decision tree has been constructed.

DTC_10_4 1x1 ClassificationTree

The below are the values for accuracy, precision, recall parameters.

accuracy_DecisionTree_10_4	0.8905
precision_D10_4	0.8615
recall_DecisionTree_10_4	0.8000

The below is the screen shot code for generating 5th random training sample. The code snippet also does the following:

- Constructing a decision tree with 10 leaf nodes.
- Calculating confusion matrix.
- Calculating the accuracy, recall, precision values.

```
Generating 5th dataset for traing data
k5 = randperm(size(Data2,1));%Creating a random permutation vector
Data2_training_5 = Data2 (k5(1: 210), :);%Dividing the dataset2 into 210 training rows

load Data2_training_5
DTC_10_5=fitctree(Data2_training_5, 'class','MinLeafSize',10);%Decision tree of 10 nodes with 3rd training data

%Confusion Matrics generation with variables with 3rd training data
Actual_D_5 = Data2_training_5{1:210,{'class'}};
prediction_10_5=predict(DTC_10_5,Data2_training_5);

%Confusion Matrix for Decision Tree with 10 nodes
C10_5=confusionmat(Actual_D_5, prediction_10_5);

%Calculating recall performances with seccond training data
recall_DecisionTree_10_5=C10_5(2,2)/(C10_5(2,2)+C10_5(2,1));


%Calculating accuracy performances
accuracy_DecisionTree_10_5=(C10_5(1,1)+C10_5(2,2))/(C10_5(1,1)+C10_5(1,2)+C10_5(2,1)+C10_5(2,2));

%Calculating percision performances

precision_D10_5=C10_5(2,2)/(C10_5(2,2)+C10_5(1,2));
```

The training data has been named as the Data2_training_5.

Data2_training_5	*210x7 table*

The decision tree has been constructed.

DTC_10_5	*1x1 ClassificationTree*

The accuracy, recall, performance parameters have been shown in the below snippets.

accuracy_DecisionTree_10_5	0.9000
precision_D10_5	0.8533
recall_DecisionTree_10_5	0.8649

The below screenshot shows the average and Standard Deviations of the accuracies, precision, recall values of the 5 decision trees which have been constructed with the 5 training datasets.
Note: For space reasons, I have shown only the SD for accuracy in the below screenshot.

```
%Calculating Averages, percision, recall's of the performance measures
sum_accuracy=0;
sum=0;
sum_accuracy=(accuracy_DecisionTree_10_1+accuracy_DecisionTree_10_2);
sum=sum_accuracy+accuracy_DecisionTree_10_3+accuracy_DecisionTree_10_4+accuracy_DecisionTree_10_5;
Avg_accuracy=0;
Avg_accuracy=sum/5;


sum_percision=0;
Avg_percision=0;
sum_percision=(precision_D10_1+precision_D10_2+precision_D10_3+precision_D10_4+precision_D10_5);
Avg_percision=sum_percision/5;

sum_reacll=0;
Avg_recall=0;
sum_reacll=(recall_DecisionTree_10_1+recall_DecisionTree_10_2+recall_DecisionTree_10_3+recall_DecisionTree_10_4+rec
Avg_recall=sum_reacll/5;

%Calculating SD's of the performance measures
Accuracy_D10=[accuracy_DecisionTree_10_1 accuracy_DecisionTree_10_2 accuracy_DecisionTree_10_3 accuracy_DecisionTre
std(Accuracy_D10)
```

The average accuracy of the 5 decision trees constructed is around 0.8905. When we compare this to the accuracy of the decision tree (0.8714) which we generated at the beginning, we find that on generating the trees, the average accuracy increases.

The average precision is around 0.8435 when compared to the first generated precision value which is 0.8269. We can say that the precision value decreased and not sure if the predicted data accurately predicted the actual values.

The average recall is approx. 0.8202 which is far more significant than the first generated recall value with the original training data set which is 0.749.

Avg_accuracy	0.8905
Avg_percision	0.8435
Avg_recall	0.8202

The standard deviations of the decision tree parameters are shown below.

SD_Accuracy	0.0178
SD_Percision	0.0223
SD_Recall	0.0431

2. Repeat the same tasks as done in Question-1 above for Data3. In addition to reporting results for 2a, 2b, and 2c, comment on the comparison of results obtained for 1c and 2c. Give your analysis for the differences in results. Label this answer as 2d.

Answer:

I have imported the data using the Import Button.

For space reasons, I am showing only a partial amount of the dataset.

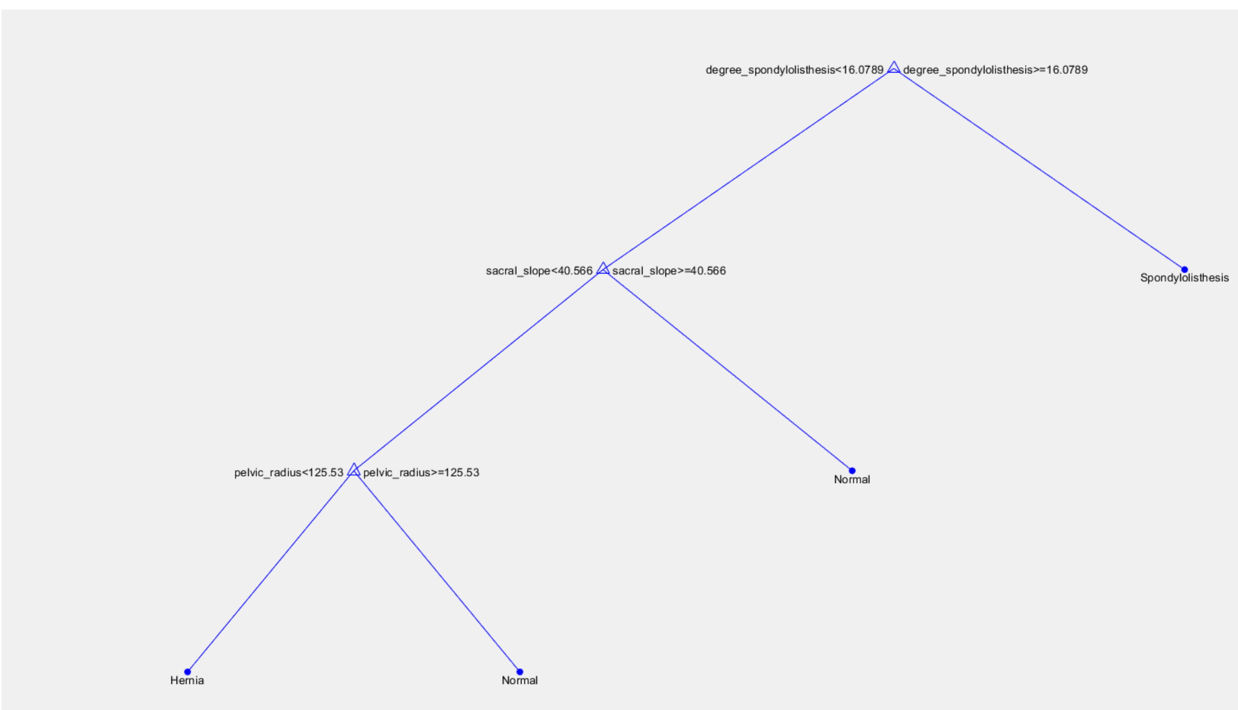
1	2	3	4	5	6	7	8
pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis	class	
48.1092	14.9307	35.5647	33.1785	124.0565	7.9479	Hernia	
38.5053	16.9643	35.1128	21.5410	127.6329	7.9867	Normal	
78.4917	22.1818	60.0000	56.3099	118.5303	27.3832	Spondylolis...	
92.0263	35.3927	77.4170	56.6336	115.7235	58.0575	Spondylolis...	
65.7563	13.2069	44.0000	52.5494	129.3936	-1.9821	Normal	
56.4470	19.4445	43.5778	37.0025	139.1897	-1.8597	Normal	
64.2615	14.4979	43.9025	49.7636	115.3883	5.9515	Normal	
64.2748	12.5086	68.7024	51.7662	95.2525	39.4098	Spondylolis...	
31.2324	17.7158	15.5000	13.5166	120.0554	0.4998	Hernia	
56.0302	16.2979	62.2753	39.7323	114.0231	-2.3257	Hernia	
72.9556	19.5770	61.0071	53.3787	111.2340	0.8135	Normal	
44.9367	17.4438	27.7806	27.4928	117.9803	5.5696	Hernia	
61.8216	13.5971	64.0000	48.2245	121.7798	1.2962	Normal	
40.7470	1.8355	50.0000	38.9115	139.2472	0.6686	Normal	
69.0049	13.2918	55.5701	55.7131	126.6116	10.8320	Normal	
48.2599	16.4175	36.3291	31.8425	94.8823	28.3438	Spondylolis...	
89.5049	48.9037	72.0034	40.6013	134.6343	118.3534	Spondylolis...	
54.7418	12.0951	41.0000	42.6467	117.6432	40.3823	Spondylolis...	
26.1479	10.7595	14	15.3885	125.2033	-10.0931	Hernia	
31.2760	3.1447	32.5630	28.1313	129.0114	3.6230	Hernia	
60.6262	20.5960	64.5353	40.0303	117.2256	104.8592	Spondylolis...	
66.8792	24.8920	49.2786	41.9872	113.4770	-2.0059	Hernia	
57.1459	16.4891	42.8421	40.6568	113.8062	5.0152	Normal	
89.8347	22.6392	90.5635	67.1955	100.5012	3.0410	Normal	
77.1213	30.3499	77.4811	46.7715	110.6111	82.0936	Spondylolis...	
63.0736	24.4138	54.0000	38.6598	106.4243	15.7797	Hernia	
49.8281	16.7364	28	33.0917	121.4356	1.9133	Normal	
65.0138	9.8383	57.7358	55.1755	94.7385	49.6970	Spondylolis...	
77.2369	16.7376	49.7755	60.4993	110.6904	39.7872	Spondylolis...	
50.7533	20.2351	37	30.5182	122.3435	2.2885	Normal	
74.4336	41.5573	27.7000	32.8763	107.9493	5.0001	Hernia	
80.1116	33.9424	85.1016	46.1691	125.5936	100.2921	Spondylolis...	
48.3189	17.4521	48.0000	30.8668	128.9803	-0.9109	Normal	
43.9228	14.1780	37.8325	29.7449	134.4610	6.4516	Hernia	
59.7261	7.7249	55.3435	52.0013	125.1742	3.2352	Normal	

The below screenshot is the Matlab code for creation of the decision trees with 5, 10,15,20,25 leaf node; I have used the function fitctree to generate the decision tree using the arguments such as MinLeafSize as 5, 10,15,20,25 to limit leaf nodes.

```
r = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training = Data3 (r(1: 210), :);%Dividing the dataset2 into 210 training rows
%Data2_test = Data2 (k(211: end), :);%Dividing the dataset2 into 100 test rows


load Data3_training
D3TC_5=fitctree(Data3_training, 'class','MinLeafSize',5);
%Decision Trees Constructing
D3TC_10=fitctree(Data3_training, 'class','MinLeafSize',10);
D3TC_15=fitctree(Data3_training, 'class','MinLeafSize',15);
D3TC_20=fitctree(Data3_training, 'class','MinLeafSize',20);
D3TC_25=fitctree(Data3_training, 'class','MinLeafSize',25);
```

2a.
Answer:
The below is the screen shot for the tree with 25 leaf nodes.
I have noticed that the as long the leaf node increases the tree height decreases and the child nodes too.



2b.
Answer:
For calculating the performance parameters, I have computed the confusion matrix and calculated the accuracy, precision, recall values.

Accuracy:

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.
Accuracy = TP+TN/TP+FP+FN+TN

Precision:

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.
Precision = TP/TP+FP

Recall:
Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

Recall = TP/TP+FN

I have computed the confusion matrices using the confusionmat function and calculated the Accuracy, Recall, and Precision using the formula as shown in the screenshot.

```
%---------------------------------Q1.2b------------------------%
%Confusion Matrics
Actual_D3 = Data3_training{1:210,{'class'}};
%Confusion Matrix for Decision Tree with 5 nodes
prediction_D3_5=predict(D3TC_5,Data3_training);
C5_D3=confusionmat(Actual_D3, prediction_D3_5);

%Confusion Matrix for Decision Tree with 10 nodes
prediction__D3_10=predict(D3TC_10,Data3_training);
C10_D3=confusionmat(Actual_D3, prediction__D3_10);

%Confusion Matrix for Decision Tree with 15 nodes
prediction__D3_15=predict(D3TC_15,Data3_training);
C15_D3=confusionmat(Actual_D3, prediction__D3_15);

%Confusion Matrix for Decision Tree with 20 nodes
prediction__D3_20=predict(D3TC_20,Data3_training);
C20_D3=confusionmat(Actual_D3, prediction__D3_20);
%[c_matrixp,Result]= confusion.getMatrix(Actual_D3,prediction_D3_20);


%Confusion Matrix for Decision Tree with 25 nodes
prediction_D3_25=predict(D3TC_25,Data3_training);
C25_D3=confusionmat(Actual_D3, prediction_D3_25);
```

I have calculated the performance parameters using the formula's and below is the screenshot for the calculation of the accuracy, recall, performance.

```
%Calculating recall performances

recall_Decision3Tree_5_Normal=C5_D3(2,2)/(C5_D3(2,2)+C5_D3(2,1)+C5_D3(2,3));
recall_Decision3Tree_10_Normal=C10_D3(2,2)/(C10_D3(2,2)+C10_D3(2,1)+C10_D3(2,3));
recall_Decision3Tree_15_Normal=C15_D3(2,2)/(C15_D3(2,2)+C15_D3(2,1)+C15_D3(2,3));
recall_Decision3Tree_20_Normal=C20_D3(2,2)/(C20_D3(2,2)+C20_D3(2,1)+C20_D3(2,3));
recall_Decision3Tree_25_Normal=C25_D3(2,2)/(C25_D3(2,2)+C25_D3(2,1)+C25_D3(2,3));


%Calculating accuracy performances
accuracy_DecisionTree_5=(C5_D3(1,1)+C5_D3(2,2)+C5_D3(3,3))/(C5_D3(1,1)+C5_D3(1,2)+C5_D3(2,1)+C5_D3(2,2)+C5_D3(3,3)+
accuracy_DecisionTree_10=(C10_D3(1,1)+C10_D3(2,2)+C10_D3(3,3))/(C10_D3(1,1)+C10_D3(1,2)+C10_D3(2,1)+C10_D3(2,2)+C10
accuracy_DecisionTree_15=(C15_D3(1,1)+C15_D3(2,2)+C15_D3(3,3))/(C15_D3(1,1)+C15_D3(1,2)+C15_D3(2,1)+C15_D3(2,2)+C15
accuracy_DecisionTree_20=(C20_D3(1,1)+C20_D3(2,2)+C20_D3(3,3))/(C20_D3(1,1)+C20_D3(1,2)+C20_D3(2,1)+C20_D3(2,2)+C20
accuracy_DecisionTree_25=(C25_D3(1,1)+C25_D3(2,2)+C25_D3(3,3))/(C25_D3(1,1)+C25_D3(1,2)+C25_D3(2,1)+C25_D3(2,2)+C25

%Calculating percision performances

precision_D5_normal=C5_D3(2,2)/(C5_D3(2,2)+C5_D3(1,2)+C5_D3(3,2));
precision_D10_normal=C10_D3(2,2)/(C10_D3(2,2)+C10_D3(1,2)+C10_D3(3,2));
precision_D15_normal=C15_D3(2,2)/(C15_D3(2,2)+C15_D3(1,2)+C15_D3(3,2));
precision_D20_normal=C20_D3(2,2)/(C20_D3(2,2)+C20_D3(1,2)+C20_D3(3,2));
precision_D25_normal=C25_D3(2,2)/(C25_D3(2,2)+C25_D3(1,2)+C25_D3(3,2));
```

The below screenshot represents the performance values of the Decision tree.

Decision tree with leaf nodes 5 has the highest precision value whereas decision tree with leaf nodes 15 has the least precision value.

precision_D15_normal	0.7857
precision_D20_normal	0.8364
precision_D25_normal	0.8305
precision_D5_normal	0.9016
precision_D10_normal	0.8182

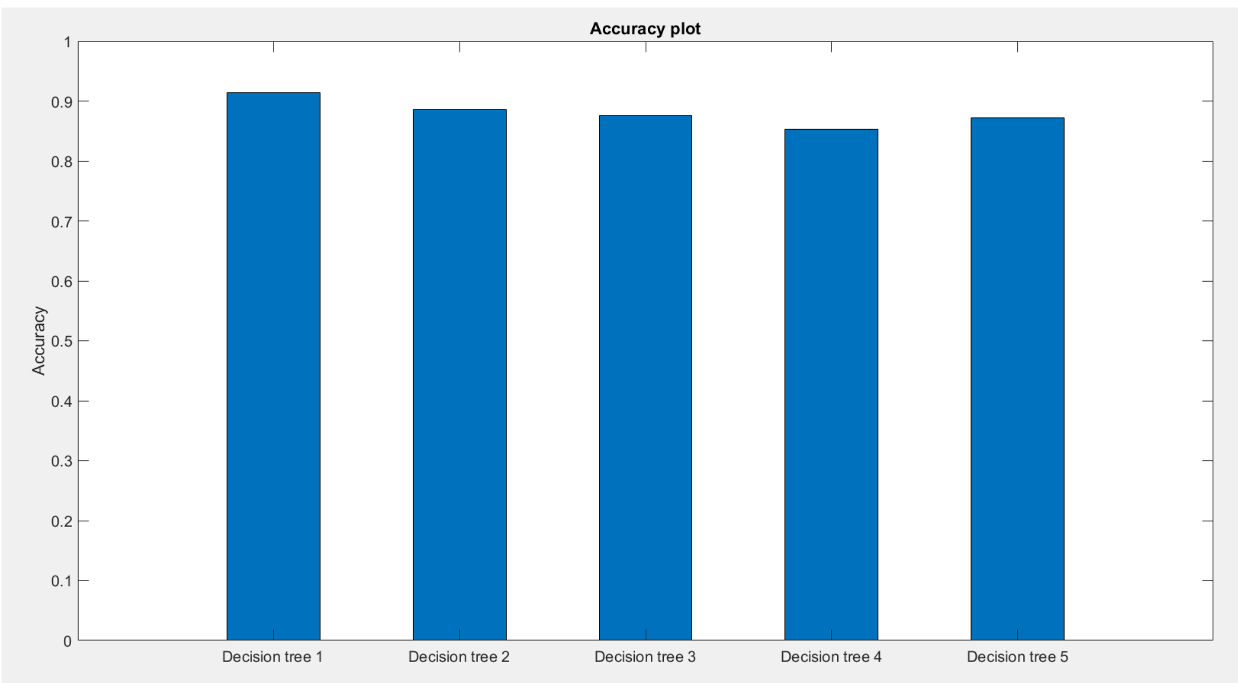
The accuracy is high in the case of the decision tree with the leaf nodes 5 and least in the case of the decision tree with 20 nodes.

accuracy_DecisionTree_15	0.8762
accuracy_DecisionTree_20	0.8524
accuracy_DecisionTree_25	0.8714
accuracy_DecisionTree_5	0.9143
accuracy_DecisionTree_10	0.8857

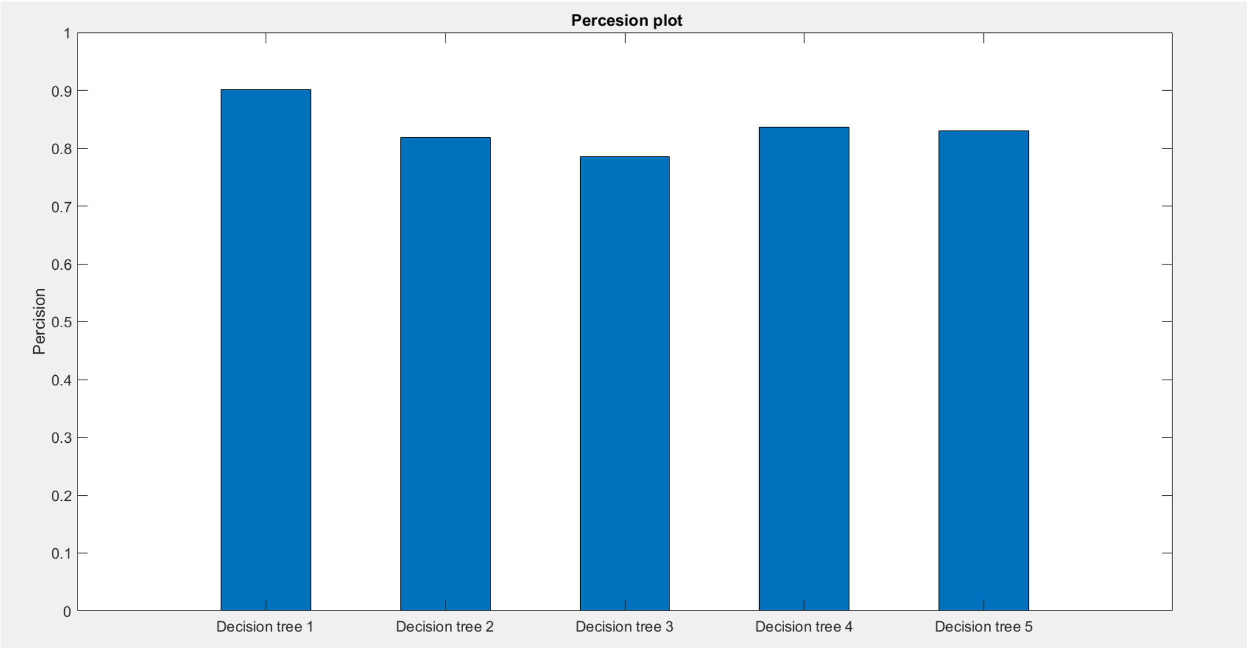
The recall is high in the two cases of the decision tree with 5 and 15 leaf nodes. Decision tree with leaf nodes 20 has the least recall value.

recall_Decision3Tree_15_Normal	0.8333
recall_Decision3Tree_20_Normal	0.6970
recall_Decision3Tree_25_Normal	0.7424
recall_Decision3Tree_5_Normal	0.8333
recall_Decision3Tree_10_Normal	0.8182

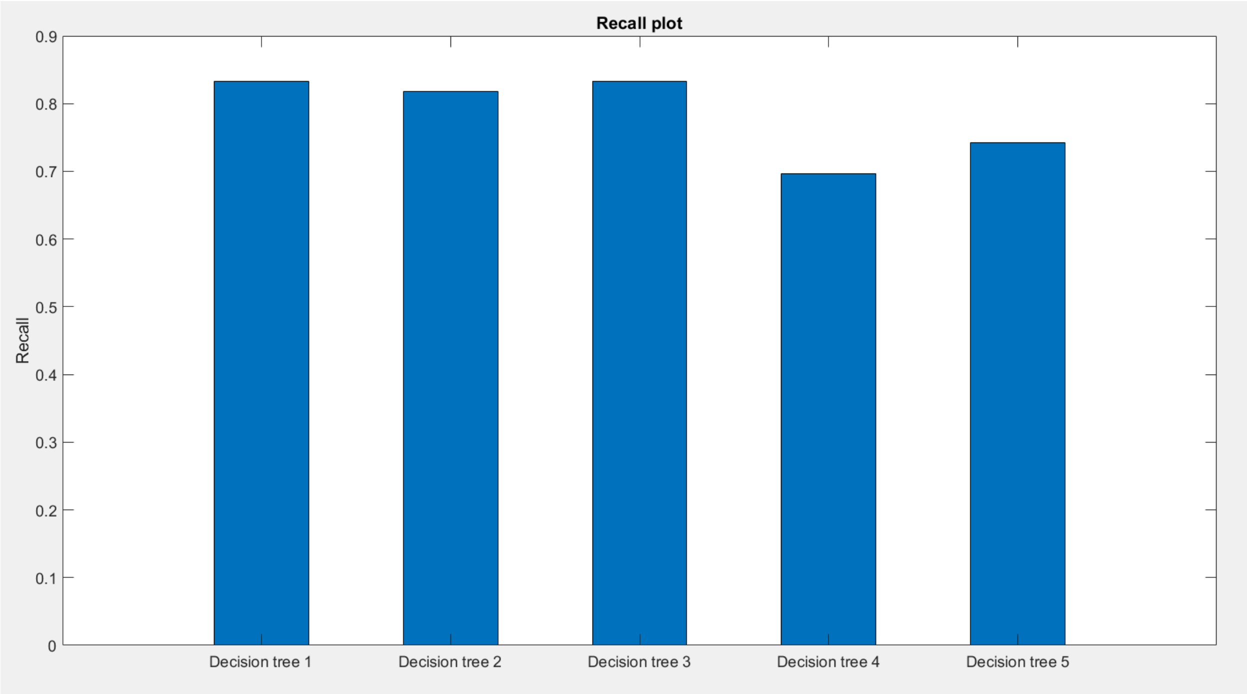
The below screenshot is for the accuracy plot. X axis represent the decision tress with the leaf nodes 5 to 25. Y axis represent the accuracy. As we can see Decision tree with 5 nodes has the highest accuracy value.



The below screenshot is for the precision plot. X axis represent the decision tress with the leaf nodes 5 to 25. Y axis represent the precision.
As we can see Decision tree with 5 nodes has the highest precision value.



The below screenshot is for the recall plot. X axis represent the decision tress with the leaf nodes 5 to 25. Y axis represent the recall.



2c.

Answer:

The below is the screen shot code for generating 5 different random training samples. The code snippet also does the following:

- Constructing a decision tree with 10 leaf nodes.
- Calculating confusion matrix.
- Calculating the accuracy, recall, precision values.

```
%Generating 5 datasets

r1 = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training_1 = Data3 (r1(1: 210), :);%Dividing the dataset2 into 210 training rows

load Data3_training_1
D3TC_10_1=fitctree(Data3_training_1, 'class','MinLeafSize',10);

Actual_D3_1 = Data3_training_1{1:210,{'class'}};
prediction__D3_10_1=predict(D3TC_10_1,Data3_training_1);
C10_D3_1=confusionmat(Actual_D3_1, prediction__D3_10_1);

recall_Decision3Tree_10_Normal_1=C10_D3_1(2,2)/(C10_D3_1(2,2)+C10_D3_1(2,1)+C10_D3_1(2,3));

accuracy_DecisionTree_10_1=(C10_D3_1(1,1)+C10_D3_1(2,2)+C10_D3_1(3,3))/(C10_D3_1(1,1)+C10_D3_1(1,2)+C10_D3_1(2,1)+C

precision_D10_normal_1=C10_D3_1(2,2)/(C10_D3_1(2,2)+C10_D3_1(1,2)+C10_D3_1(3,2));


r2 = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training_2 = Data3 (r2(1: 210), :);%Dividing the dataset2 into 210 training rows

load Data3_training_2
D3TC_10_2=fitctree(Data3_training_2, 'class','MinLeafSize',10);

Actual_D3_2 = Data3_training_2{1:210,{'class'}};
prediction__D3_10_2=predict(D3TC_10_2,Data3_training_2);
C10_D3_2=confusionmat(Actual_D3_2, prediction__D3_10_2);

recall_Decision3Tree_10_Normal_2=C10_D3_2(2,2)/(C10_D3_2(2,2)+C10_D3_2(2,1)+C10_D3_2(2,3));

accuracy_DecisionTree_10_2=(C10_D3_2(1,1)+C10_D3_2(2,2)+C10_D3_2(3,3))/(C10_D3_2(1,1)+C10_D3_2(1,2)+C10_D3_2(2,1)+C

precision_D10_normal_2=C10_D3_2(2,2)/(C10_D3_2(2,2)+C10_D3_2(1,2)+C10_D3_2(3,2));

r3 = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training_3 = Data3 (r3(1: 210), :);%Dividing the dataset2 into 210 training rows

load Data3_training_3
D3TC_10_3=fitctree(Data3_training_3, 'class','MinLeafSize',10);

Actual_D3_3 = Data3_training_3{1:210,{'class'}};
prediction__D3_10_3=predict(D3TC_10_3,Data3_training_3);
C10_D3_3=confusionmat(Actual_D3_3, prediction__D3_10_3);

recall_Decision3Tree_10_Normal_3=C10_D3_3(2,2)/(C10_D3_3(2,2)+C10_D3_3(2,1)+C10_D3_3(2,3));

accuracy_DecisionTree_10_3=(C10_D3_3(1,1)+C10_D3_3(2,2)+C10_D3_3(3,3))/(C10_D3_3(1,1)+C10_D3_3(1,2)+C10_D3_3(2,1)+C

precision_D10_normal_3=C10_D3_3(2,2)/(C10_D3_3(2,2)+C10_D3_3(1,2)+C10_D3_3(3,2));
```

```
r4 = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training_4 = Data3 (r4(1: 210), :);%Dividing the dataset2 into 210 training rows

load Data3_training_4
D3TC_10_4=fitctree(Data3_training_4, 'class','MinLeafSize',10);

Actual_D3_4 = Data3_training_4{1:210,{'class'}};
prediction__D3_10_4=predict(D3TC_10_4,Data3_training_4);
C10_D3_4=confusionmat(Actual_D3_4, prediction__D3_10_4);

recall_Decision3Tree_10_Normal_4=C10_D3_4(2,2)/(C10_D3_4(2,2)+C10_D3_4(2,1)+C10_D3_4(2,3));

accuracy_DecisionTree_10_4=(C10_D3_4(1,1)+C10_D3_4(2,2)+C10_D3_4(3,3))/(C10_D3_4(1,1)+C10_D3_4(1,2)+C10_D3_4(2,1)

precision_D10_normal_4=C10_D3_4(2,2)/(C10_D3_4(2,2)+C10_D3_4(1,2)+C10_D3_4(3,2));
```

```
r5 = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training_5 = Data3 (r5(1: 210), :);%Dividing the dataset2 into 210 training rows

load Data3_training_5
D3TC_10_5=fitctree(Data3_training_5, 'class','MinLeafSize',10);

Actual_D3_5 = Data3_training_5{1:210,{'class'}};
prediction__D3_10_5=predict(D3TC_10_5,Data3_training_5);
C10_D3_5=confusionmat(Actual_D3_5, prediction__D3_10_5);

recall_Decision3Tree_10_Normal_5=C10_D3_5(2,2)/(C10_D3_5(2,2)+C10_D3_5(2,1)+C10_D3_5(2,3));

accuracy_DecisionTree_10_5=(C10_D3_5(1,1)+C10_D3_5(2,2)+C10_D3_5(3,3))/(C10_D3_5(1,1)+C10_D3_5(1,2)+C10_D3_5(2,

precision_D10_normal_5=C10_D3_5(2,2)/(C10_D3_5(2,2)+C10_D3_5(1,2)+C10_D3_5(3,2));
```

The below are the 5 data tables which are generated after randomizing the data.
For naming conventions, I have named them as Data3_training_1 to Data3_training_5.

Data3_training_1	*210x7 table*
Data3_training_2	*210x7 table*
Data3_training_3	*210x7 table*
Data3_training_4	*210x7 table*
Data3_training_5	*210x7 table*

The below screenshot is for the accuracies of the 5 decision trees which are generated. We can see that the Decision tree which is generated with the 2nd training data has the highest accuracy rate.

accuracy_DecisionTree_10_1	0.8714
accuracy_DecisionTree_10_2	0.9000
accuracy_DecisionTree_10_3	0.8810
accuracy_DecisionTree_10_4	0.8762
accuracy_DecisionTree_10_5	0.8762

The below is the screenshot for the precision. The decision tree which is generated with the 2nd training data has the highest precision.

precision_D10_normal_1	0.8793
precision_D10_normal_2	0.8841
precision_D10_normal_3	0.8143
precision_D10_normal_4	0.8030
precision_D10_normal_5	0.7333

The below is the screen shot for the recall values. Decision tree which is generated with the 5th training data has the highest recall value.

recall_Decision3Tree_10_Normal_1	0.7391
recall_Decision3Tree_10_Normal_2	0.8356
recall_Decision3Tree_10_Normal_3	0.8382
recall_Decision3Tree_10_Normal_4	0.8154
recall_Decision3Tree_10_Normal_5	0.9016

The below screen shot represents the code for calculating the average of the performance parameters and Standard deviations of the parameters.

```
sum_accuracy_D5=accuracy_DecisionTree_10_1+accuracy_DecisionTree_10_2+accuracy_DecisionTree_10_3+accuracy_DecisionT
Avg_accuracy_D5=sum_accuracy_D5/5;

sum_percision_D5=precision_D10_normal_1+precision_D10_normal_2+precision_D10_normal_3+precision_D10_normal_4+precis
Avg_percision_D5=sum_percision_D5/5;

sum_recall_D5=recall_Decision3Tree_10_Normal_1+recall_Decision3Tree_10_Normal_2+recall_Decision3Tree_10_Normal_3+re
Avg_recall_D5=sum_recall_D5/5;


Accuracy_D5=[accuracy_DecisionTree_10_1 accuracy_DecisionTree_10_2 accuracy_DecisionTree_10_3 accuracy_DecisionTree
percision_D5=[precision_D10_normal_1 precision_D10_normal_2 precision_D10_normal_3 precision_D10_normal_4 precision
recall_D5=[recall_Decision3Tree_10_Normal_1 recall_Decision3Tree_10_Normal_2 recall_Decision3Tree_10_Normal_3 recal

SD_Accuracy_D5=std(Accuracy_D5);
SD_Recall_D5=std(recall_D5);
SD_percision=std(percision_D5);
```

The below is the screen shot for the SD's of the performance parameters.

SD_Accuracy_D5	0.0112
SD_percision	0.0621
SD_Recall_D5	0.0584

The below is the screen shot for the Average of the performance parameters.

Avg_accuracy_D5	0.8810
Avg_percision_D5	0.8228
Avg_recall_D5	0.8260

2d. Comparing the results which are obtained in 1c and 2c we find that the average accuracy for the Data2 is 0.8905 whereas the average accuracy for the Data3 is 0.8810.

We also find the precision is greater with the Data2 rather than the Data3.

But, this is not the case for the recall value.
We find that it is higher in the case of the Data3 rather than Data2.

3. Take Data2 for this question. Partition each column into four sets of equal widths of values. Assign these intervals as values 0, 1, 2, and 3 and replace each value by its corresponding interval value.

Answer:
The below is the screenshot for the code which replaces the values of the attributes with 0,1,2,3 based on their interval range.

```
%Replacing the dataset with 0,1,2,3
load Data2
 sorted_data=Data2(:,1:6);
 full_data = Data2;
 A = sorted_data{:,{'pelvic_incidence'}};
 temp_array = table2array(sorted_data);
 for jj=1:6
     minimum_value=min(temp_array(:,jj));
     maximum_value=max(temp_array(:,jj));
    interval_value=(maximum_value-minimum_value)/4;

    %Replacing the values with 0's and 1's;
    for ii=1:height(sorted_data)
         if ge(temp_array(ii,jj),minimum_value) && le(temp_array(ii,jj),minimum_value+interval_value)
            temp_array(ii,jj) = 0;
         elseif ge(temp_array(ii,jj), minimum_value+interval_value) && le(temp_array(ii,jj), minimum_value+ (2*inte
           temp_array(ii,jj) = 1;
        elseif ge(temp_array(ii,jj), minimum_value+ (2*interval_value)) && le(temp_array(ii,jj), minimum_value+ (3*
             temp_array(ii,jj) = 2;
        else
            temp_array(ii,jj) = 3;
        end
    end
 end
```

The below is the screen shot for the converted data based on their interval range.

1	2	3	4	5	6	7	8	9
pelvic_incidence	pelvic_tiltnumeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis	class		
1	2	0	1	1	0	Abnormal		
0	1	0	0	1	0	Abnormal		
1	2	1	1	1	0	Abnormal		
1	2	1	1	1	0	Abnormal		
0	1	0	0	1	0	Abnormal		
0	1	0	0	2	0	Abnormal		
1	1	0	0	2	0	Abnormal		
0	1	0	0	2	0	Abnormal		
0	1	1	0	2	0	Abnormal		
0	0	1	0	0	0	Abnormal		
0	1	0	0	1	0	Abnormal		
0	1	0	0	2	0	Abnormal		
0	1	0	0	2	0	Abnormal		
1	1	0	0	1	0	Abnormal		
1	2	1	0	2	0	Abnormal		
0	1	0	0	2	0	Abnormal		
1	1	1	1	1	0	Abnormal		
0	0	0	0	2	0	Abnormal		
0	1	0	0	2	0	Abnormal		
0	1	0	0	2	0	Abnormal		
0	1	0	0	2	0	Abnormal		
1	1	1	0	2	0	Abnormal		
1	2	1	0	1	0	Abnormal		
0	1	0	0	2	0	Abnormal		
0	2	0	0	1	0	Abnormal		
1	2	0	0	2	0	Abnormal		
0	1	0	0	2	0	Abnormal		
0	1	1	0	1	0	Abnormal		
0	2	0	0	1	0	Abnormal		
1	2	1	1	1	0	Abnormal		
0	1	1	0	1	0	Abnormal		
0	1	0	0	1	0	Abnormal		
0	1	0	0	2	0	Abnormal		
0	1	0	0	2	0	Abnormal		
1	2	1	0	2	0	Abnormal		

a. Show the boundaries for each interval for each attribute.

The below is the screenshot for the interval boundaries.
Column 1 represents the interval boundaries for the attribute 1.
Column 2 represents the interval boundaries for the attribute 2.
Column 3 represents the interval boundaries for the attribute 3.
Column 4 represents the interval boundaries for the attribute 4.
Column 5 represents the interval boundaries for the attribute 5.
Column 6 represents the interval boundaries for the attribute 6.

1	2	3	4	5	6
52.0695	7.4418	41.9356	40.3826	93.3297	96.3421
77.9910	21.4385	69.8712	67.3982	116.5768	203.7425
103.9125	35.4352	97.8068	94.4139	139.8239	311.1428
129.8340	49.4319	125.7424	121.4296	163.0710	418.5431

c. Learn a decision tree with this transformed data and compute performance parameters in the same way as done for 1c and 2c.

Answer:
The below is the screen shots of code are for generating 5 different random training samples. The code snippet also does the following:

- Constructing a decision tree with 10 leaf nodes.
- Calculating confusion matrix.
- Calculating the accuracy, recall, precision values.

```
 %Creating 5 new traing sets with the new data

 p = randperm(size(full_data,1));%Creating a random permutation vector
 full_data_training_1 = full_data (p(1: 210), :);%Dividing the dataset2 into 210 training rows

%First DT with the new data
load full_data_training_1
FDTC_10_1=fitctree(full_data_training_1, 'class','MinLeafSize',10);%Decision tree of 10 nodes with 1st training dat

%Confusion Matrics generation with variables with 1st training data
Actual_FD_1 = full_data_training_1{1:210,{'class'}};
prediction_FD_10_1=predict(FDTC_10_1,full_data_training_1);
%Confusion Matrix for Decision Tree with 10 nodes
FD_C10_1=confusionmat(Actual_FD_1, prediction_FD_10_1);
%Calculating recall performances with seccond training data
recall_FDecisionTree_10_1=FD_C10_1(2,2)/(FD_C10_1(2,2)+FD_C10_1(2,1));
%Calculating accuracy performances
accuracy_FDecisionTree_10_1=(FD_C10_1(1,1)+FD_C10_1(2,2))/(FD_C10_1(1,1)+FD_C10_1(1,2)+FD_C10_1(2,1)+FD_C10_1(2,2))
%Calculating percision performances
precision_FD10_1=FD_C10_1(2,2)/(FD_C10_1(2,2)+FD_C10_1(1,2));
```

```
Second DT with the new data
p2 = randperm(size(full_data,1));%Creating a random permutation vector
full_data_training_2 = full_data (p2(1: 210), :);%Dividing the dataset2 into 210 training rows

load full_data_training_2
FDTC_10_2=fitctree(full_data_training_2, 'class','MinLeafSize',10);%Decision tree of 10 nodes with 1st training dat

%Confusion Matrics generation with variables with 1st training data
Actual_FD_2 = full_data_training_2{1:210,{'class'}};
prediction_FD_10_2=predict(FDTC_10_2,full_data_training_2);

%Confusion Matrix for Decision Tree with 10 nodes
FD_C10_2=confusionmat(Actual_FD_2, prediction_FD_10_2);

%Calculating recall performances with seccond training data
recall_FDecisionTree_10_2=FD_C10_2(2,2)/(FD_C10_2(2,2)+FD_C10_2(2,1));


%Calculating accuracy performances
accuracy_FDecisionTree_10_2=(FD_C10_2(1,1)+FD_C10_2(2,2))/(FD_C10_2(1,1)+FD_C10_2(1,2)+FD_C10_2(2,1)+FD_C10_2(2,2))

%Calculating percision performances

precision_FD10_2=FD_C10_2(2,2)/(FD_C10_2(2,2)+FD_C10_2(1,2));
```

```
load full_data_training_3
FDTC_10_3=fitctree(full_data_training_3, 'class','MinLeafSize',10);%Decision tree of 10 nodes with 1st training data

%Confusion Matrics generation with variables with 1st training data
Actual_FD_3 = full_data_training_3{1:210,{'class'}};
prediction_FD_10_3=predict(FDTC_10_3,full_data_training_3);

%Confusion Matrix for Decision Tree with 10 nodes
FD_C10_3=confusionmat(Actual_FD_3, prediction_FD_10_3);

%Calculating recall performances with seccond training data
recall_FDecisionTree_10_3=FD_C10_3(2,2)/(FD_C10_3(2,2)+FD_C10_3(2,1));


%Calculating accuracy performances
accuracy_FDecisionTree_10_3=(FD_C10_3(1,1)+FD_C10_3(2,2))/(FD_C10_3(1,1)+FD_C10_3(1,2)+FD_C10_3(2,1)+FD_C10_3(2,2));

%Calculating percision performances
precision_FD10_3=FD_C10_3(2,2)/(FD_C10_3(2,2)+FD_C10_3(1,2));
```

```
%Fourth DT with the new data

 p4 = randperm(size(full_data,1));%Creating a random permutation vector
 full_data_training_4 = full_data (p4(1: 210), :);%Dividing the dataset2 into 210 training rows

load full_data_training_4
FDTC_10_4=fitctree(full_data_training_4, 'class','MinLeafSize',10);%Decision tree of 10 nodes with 1st training da

%Confusion Matrics generation with variables with 1st training data
Actual_FD_4 = full_data_training_4{1:210,{'class'}};
prediction_FD_10_4=predict(FDTC_10_4,full_data_training_4);

%Confusion Matrix for Decision Tree with 10 nodes
FD_C10_4=confusionmat(Actual_FD_4, prediction_FD_10_4);

%Calculating recall performances with seccond training data
recall_FDecisionTree_10_4=FD_C10_4(2,2)/(FD_C10_4(2,2)+FD_C10_4(2,1));


%Calculating accuracy performances
accuracy_FDecisionTree_10_4=(FD_C10_4(1,1)+FD_C10_4(2,2))/(FD_C10_4(1,1)+FD_C10_4(1,2)+FD_C10_4(2,1)+FD_C10_4(2,2)

%Calculating percision performances

precision_FD10_4=FD_C10_4(2,2)/(FD_C10_4(2,2)+FD_C10_4(1,2));
```

```
%5th DT with the new data

 p5 = randperm(size(full_data,1));%Creating a random permutation vector
full_data_training_5 = full_data (p5(1: 210), :);%Dividing the dataset2 into 210 training rows

load full_data_training_5
FDTC_10_5=fitctree(full_data_training_5, 'class','MinLeafSize',10);%Decision tree of 10 nodes with 1st training dat

%Confusion Matrics generation with variables with 1st training data
Actual_FD_5 = full_data_training_5{1:210,{'class'}};
prediction_FD_10_5=predict(FDTC_10_5,full_data_training_5);

%Confusion Matrix for Decision Tree with 10 nodes
FD_C10_5=confusionmat(Actual_FD_5, prediction_FD_10_5);

%Calculating recall performances with seccond training data
recall_FDecisionTree_10_5=FD_C10_5(1,1)/(FD_C10_5(2,2)+FD_C10_5(2,1));


%Calculating accuracy performances
accuracy_FDecisionTree_10_5=(FD_C10_5(1,1)+FD_C10_5(2,2))/(FD_C10_5(1,1)+FD_C10_5(1,2)+FD_C10_5(2,1)+FD_C10_5(2,2))

%Calculating percision performances

precision_FD10_5=FD_C10_5(2,2)/(FD_C10_5(2,2)+FD_C10_5(1,2));
```

For naming conventions, I have named the 5 generated training data as full_data_training_1 to full_data_training_5.

full_data_training_1	*210x7 table*
full_data_training_2	*210x7 table*
full_data_training_3	*210x7 table*
full_data_training_4	*210x7 table*
full_data_training_5	*210x7 table*

The below screenshot represents the accuracies of the five different decision trees which have been constructed and we find that the decision tree constructed with the 3rd and 4th training data have the same accuracy.

accuracy_FDecisionTree_10_1	0.7667
accuracy_FDecisionTree_10_2	0.7476
accuracy_FDecisionTree_10_3	0.7810
accuracy_FDecisionTree_10_4	0.7810
accuracy_FDecisionTree_10_5	0.7571

From the below screenshot, we can infer that the recall for the decision tree which is constructed with the 5th training data has the highest recall value.

recall_FDecisionTree_10_1	0.5909
recall_FDecisionTree_10_2	0.7917
recall_FDecisionTree_10_3	0.5915
recall_FDecisionTree_10_4	0.8028
recall_FDecisionTree_10_5	1.8485

The below screenshot represents the precision values for the decision trees which are constructed with the 5 training data samples.

precision_FD10_1	0.6393
precision_FD10_2	0.6000
precision_FD10_3	0.7119
precision_FD10_4	0.6404
precision_FD10_5	0.6271

The below screenshot represent the code for calculating the average of the performance measures and the SD's of the performance measures.

```
%Calculating the Accuracy, percision, recall of the new data set

sum_accuracy_FD=0;
sum_FD=0;
sum_accuracy_FD=(accuracy_FDecisionTree_10_1+accuracy_FDecisionTree_10_2);
sum_FD=sum_accuracy_FD+accuracy_FDecisionTree_10_3+accuracy_FDecisionTree_10_4+accuracy_FDecisionTree_10_5;
Avg_accuracy_FD=0;
Avg_accuracy_FD=sum_FD/5;

percision_FD=0;
Avg_percision_FD=0;
sum_percision_FD=(precision_FD10_1+precision_FD10_2+precision_FD10_3+precision_FD10_4+precision_FD10_5);
Avg_percision_FD=sum_percision_FD/5;

sum_reacll_FD=0;
Avg_recall_FD=0;
sum_reacll_FD=(recall_FDecisionTree_10_1+recall_FDecisionTree_10_2+recall_FDecisionTree_10_3+recall_FDecisionTree_1

Avg_recall_FD=sum_reacll_FD/5;

%Calculating SD's of the performance measures
Accuracy_FD10=[accuracy_FDecisionTree_10_1 accuracy_FDecisionTree_10_2 accuracy_FDecisionTree_10_3 accuracy_FDecisi
std(Accuracy_FD10)

Percision_FD10=[precision_FD10_1 precision_FD10_2 precision_FD10_3 precision_FD10_4 precision_FD10_5];
std(Percision_FD10)

Recall_FD10=[recall_FDecisionTree_10_1 recall_FDecisionTree_10_2 recall_FDecisionTree_10_3 recall_FDecisionTree_10_
std(Recall_FD10)
```

The average of the performance parameters are shown below:

Avg_accuracy_FD	0.7667
Avg_percision_FD	0.6438
Avg_recall_FD	0.9251

The SD's of the performance parameters are below:

SD_FD_accuracy	0.0147
SD_FD_Percision	0.0414
SD_FD_Recall	0.5264

3c.

Compare these results with those obtained for 1c. Analyze the differences in performance and give your intuitive reasons why these differences are observed.

Answer:

We find that the values in 1c are much better than the values in the 3b. The average accuracy in 1c is around 0.89 whereas in 3b it is around 0.76.

But that's not in the case of the recall parameter. The average value of the recall parameter is in fact has increased.

Since, we are classifying the data based on their intervals, this classification and change in the dataset will have significant changes in the performance parameters.

Since, in the 1c, data2's each attributes are being taken as a whole wide one interval which has the chance that the accuracy and precision parameters are better than 3b. The opposite rule follows for the recall parameter, which is since, the data2 is divided in intervals the average recall value would be higher.

Matlab Code:

Import the dataset using the import option available in the matlab.

```
k = randperm(size(Data2,1));%Creating a random permutation vector
Data2_training = Data2 (k(1: 210), :);%Dividing the dataset2 into 210
training rows
Data2_test = Data2 (k(211: end), :);%Dividing the dataset2 into 100 test rows

%Decision Trees Constructing
load Data2_training
DTC_5=fitctree(Data2_training, 'class','MinLeafSize',3);
DTC_10=fitctree(Data2_training, 'class','MinLeafSize',10);
DTC_15=fitctree(Data2_training, 'class','MinLeafSize',15);
DTC_20=fitctree(Data2_training, 'class','MinLeafSize',20);
DTC_25=fitctree(Data2_training, 'class','MinLeafSize',25);
view((DTC_5),'mode','graph')
view((DTC_10),'mode','graph')
view((DTC_15),'mode','graph')
view((DTC_20),'mode','graph')
%%
%
%   for x = 1:10
%       disp(x)
%   end
%
view((DTC_25),'mode','graph')
%---------------------------------Q1.2b------------------------%
%Confusion Matrics
Actual_D = Data2_training{1:210,{'class'}};
%Confusion Matrix for Decision Tree with 5 nodes
prediction_5=predict(DTC_5,Data2_training);
C5=confusionmat(Actual_D, prediction_5);

%Confusion Matrix for Decision Tree with 10 nodes
prediction_10=predict(DTC_10,Data2_training);
C10=confusionmat(Actual_D, prediction_10);

%Confusion Matrix for Decision Tree with 15 nodes
prediction_15=predict(DTC_15,Data2_training);
C15=confusionmat(Actual_D, prediction_15);

%Confusion Matrix for Decision Tree with 20 nodes
prediction_20=predict(DTC_20,Data2_training);
C20=confusionmat(Actual_D, prediction_20);

%Confusion Matrix for Decision Tree with 25 nodes
prediction_25=predict(DTC_25,Data2_training);
C25=confusionmat(Actual_D, prediction_25);

%Calculating recall performances
```

```
%Calculating recall performances for Decision tree with 5 nodes
recall_DecisionTree_5=C5(2,2)/(C5(2,2)+C5(2,1))

%Calculating recall performances for Decision tree with 10 nodes
recall_DecisionTree_10=C10(2,2)/(C10(2,2)+C10(2,1))

%Calculating recall performances for Decision tree with 15 nodes
recall_DecisionTree_15=C15(2,2)/(C15(2,2)+C15(2,1))

%Calculating recall performances for Decision tree with 20 nodes
recall_DecisionTree_20=C20(2,2)/(C20(2,2)+C20(2,1))

%Calculating recall performances for Decision tree with 25 nodes
recall_DecisionTree_25=C25(2,2)/(C25(2,2)+C25(2,1))

%Calculating accuracy performances

%Calculating accuracy performances for Decision tree with 5 nodes
accuracy_DecisionTree_5=(C5(1,1)+C5(2,2))/(C5(1,1)+C5(1,2)+C5(2,1)+C5(2,2))
%Calculating accuracy performances for Decision tree with 10 nodes
accuracy_DecisionTree_10=(C10(1,1)+C10(2,2))/(C10(1,1)+C10(1,2)+C10(2,1)+C10(
2,2))
%Calculating accuracy performances for Decision tree with 15 nodes
accuracy_DecisionTree_15=(C15(1,1)+C15(2,2))/(C15(1,1)+C15(1,2)+C15(2,1)+C15(
2,2))
%Calculating accuracy performances for Decision tree with 20 nodes
accuracy_DecisionTree_20=(C20(1,1)+C20(2,2))/(C20(1,1)+C20(1,2)+C20(2,1)+C20(
2,2))
%Calculating accuracy performances for Decision tree with 25 nodes
accuracy_DecisionTree_25=(C25(1,1)+C25(2,2))/(C25(1,1)+C25(1,2)+C25(2,1)+C25(
2,2))
%Calculating percision performances

%Calculating percision performances for Decision tree with 5 nodes
precision_D5=C5(2,2)/(C5(2,2)+C5(1,2))
%Calculating percision  performances for Decision tree with 10 nodes
precision_D10=C10(2,2)/(C10(2,2)+C10(1,2))
%Calculating percision  performances for Decision tree with 15 nodes
precision_D15=C15(2,2)/(C15(2,2)+C15(1,2))
%Calculating percision  performances for Decision tree with 20 node
precision_D20=C20(2,2)/(C20(2,2)+C20(1,2))
%Calculating percision  performances for Decision tree with 25 nodes
precision_D25=C25(2,2)/(C25(2,2)+C25(1,2))

plot_accuracy=[accuracy_DecisionTree_5 accuracy_DecisionTree_10
accuracy_DecisionTree_15 accuracy_DecisionTree_20 accuracy_DecisionTree_25];
plot_recall=[recall_DecisionTree_5 recall_DecisionTree_10
recall_DecisionTree_15 recall_DecisionTree_20 recall_DecisionTree_25];

plot_percision=[precision_D5 precision_D10 precision_D15 precision_D20
precision_D25];

figure;
bar(plot_accuracy, 0.5);
ylabel('Accuracy', 'FontSize', 15);
```

```
title('Accuracy plot');
names = {'Decision tree 1', 'Decision tree 2', 'Decision tree 3', 'Decision
tree 4', 'Decision tree 5'};
set(gca, 'xticklabel', names, 'FontSize', 15);


figure;
bar(plot_recall, 0.5);
ylabel('Recall', 'FontSize', 15);
title('Recall plot');
names = {'Decision tree 1', 'Decision tree 2', 'Decision tree 3', 'Decision
tree 4', 'Decision tree 5'};
set(gca, 'xticklabel', names, 'FontSize', 15);

figure;
bar(plot_percision, 0.5);
ylabel('Percision', 'FontSize', 15);
title('Percesion plot');
names = {'Decision tree 1', 'Decision tree 2', 'Decision tree 3', 'Decision
tree 4', 'Decision tree 5'};
set(gca, 'xticklabel', names, 'FontSize', 15);


%-------------------------------Q1.3c---------------------------------------
-------%
%Generating 5 datasets

%Generating 1st dataset for traing data
k1 = randperm(size(Data2,1));%Creating a random permutation vector
Data2_training_1 = Data2 (k1(1: 210), :);%Dividing the dataset2 into 210
training rows

load Data2_training_1
DTC_10_1=fitctree(Data2_training_1, 'class','MinLeafSize',10);%Decision tree
of 10 nodes with 2nd training data

%Confusion Matrics generation with variables with 2nd training data
Actual_D_1 = Data2_training_1{1:210,{'class'}};
prediction_10_1=predict(DTC_10_1,Data2_training_1);

%Confusion Matrix for Decision Tree with 10 nodes
[C10_1,C10_1_order]=confusionmat(Actual_D_1, prediction_10_1);

%Calculating recall performances with seccond training data
recall_DecisionTree_10_1=C10_1(2,2)/(C10_1(2,2)+C10_1(2,1));


%Calculating accuracy performances
accuracy_DecisionTree_10_1=(C10_1(1,1)+C10_1(2,2))/(C10_1(1,1)+C10_1(1,2)+C10
_1(2,1)+C10_1(2,2));

%Calculating percision performances
precision_D10_1=C10_1(2,2)/(C10_1(2,2)+C10_1(1,2));
```

```
%Generating 2nd dataset for traing data
k2 = randperm(size(Data2,1));%Creating a random permutation vector
Data2_training_2 = Data2 (k2(1: 210), :);%Dividing the dataset2 into 210
training rows

load Data2_training_2
DTC_10_2=fitctree(Data2_training_2, 'class','MinLeafSize',10);%Decision tree
of 10 nodes with 3rd training data

%Confusion Matrics generation with variables with 3rd training data
Actual_D_2 = Data2_training_2{1:210,{'class'}};
prediction_10_2=predict(DTC_10_2,Data2_training_2);

%Confusion Matrix for Decision Tree with 10 nodes
C10_2=confusionmat(Actual_D_2, prediction_10_2);

%Calculating recall performances with seccond training data
recall_DecisionTree_10_2=C10_2(2,2)/(C10_2(2,2)+C10_2(2,1));


%Calculating accuracy performances
accuracy_DecisionTree_10_2=(C10_2(1,1)+C10_2(2,2))/(C10_2(1,1)+C10_2(1,2)+C10
_2(2,1)+C10_2(2,2));

%Calculating percision performances

precision_D10_2=C10_2(2,2)/(C10_2(2,2)+C10_2(1,2));


%Generating 3rd dataset for traing data
k3 = randperm(size(Data2,1));%Creating a random permutation vector
Data2_training_3 = Data2 (k3(1: 210), :);%Dividing the dataset2 into 210
training rows

load Data2_training_3
DTC_10_3=fitctree(Data2_training_3, 'class','MinLeafSize',10);%Decision tree
of 10 nodes with 3rd training data

%Confusion Matrics generation with variables with 3rd training data
Actual_D_3 = Data2_training_3{1:210,{'class'}};
prediction_10_3=predict(DTC_10_3,Data2_training_3);

%Confusion Matrix for Decision Tree with 10 nodes
C10_3=confusionmat(Actual_D_3, prediction_10_3);

%Calculating recall performances with seccond training data
recall_DecisionTree_10_3=C10_3(2,2)/(C10_3(2,2)+C10_3(2,1));


%Calculating accuracy performances
accuracy_DecisionTree_10_3=(C10_3(1,1)+C10_3(2,2))/(C10_3(1,1)+C10_3(1,2)+C10
_3(2,1)+C10_3(2,2));
```

```
%Calculating percision performances

precision_D10_3=C10_3(2,2)/(C10_3(2,2)+C10_3(1,2));


%Generating 4th dataset for traing data
k4 = randperm(size(Data2,1));%Creating a random permutation vector
Data2_training_4 = Data2 (k4(1: 210), :);%Dividing the dataset2 into 210
training rows

load Data2_training_4
DTC_10_4=fitctree(Data2_training_4, 'class','MinLeafSize',10);%Decision tree
of 10 nodes with 3rd training data

%Confusion Matrics generation with variables with 3rd training data
Actual_D_4 = Data2_training_4{1:210,{'class'}};
prediction_10_4=predict(DTC_10_4,Data2_training_4);

%Confusion Matrix for Decision Tree with 10 nodes
C10_4=confusionmat(Actual_D_4, prediction_10_4);

%Calculating recall performances with seccond training data
recall_DecisionTree_10_4=C10_4(2,2)/(C10_4(2,2)+C10_4(2,1));


%Calculating accuracy performances
accuracy_DecisionTree_10_4=(C10_4(1,1)+C10_4(2,2))/(C10_4(1,1)+C10_4(1,2)+C10
_4(2,1)+C10_4(2,2));

%Calculating percision performances

precision_D10_4=C10_4(2,2)/(C10_4(2,2)+C10_4(1,2));



%Generating 5th dataset for traing data
k5 = randperm(size(Data2,1));%Creating a random permutation vector
Data2_training_5 = Data2 (k5(1: 210), :);%Dividing the dataset2 into 210
training rows

load Data2_training_5
DTC_10_5=fitctree(Data2_training_5, 'class','MinLeafSize',10);%Decision tree
of 10 nodes with 3rd training data

%Confusion Matrics generation with variables with 3rd training data
Actual_D_5 = Data2_training_5{1:210,{'class'}};
prediction_10_5=predict(DTC_10_5,Data2_training_5);

%Confusion Matrix for Decision Tree with 10 nodes
C10_5=confusionmat(Actual_D_5, prediction_10_5);

%Calculating recall performances with seccond training data
```

```
recall_DecisionTree_10_5=C10_5(2,2)/(C10_5(2,2)+C10_5(2,1));


%Calculating accuracy performances
accuracy_DecisionTree_10_5=(C10_5(1,1)+C10_5(2,2))/(C10_5(1,1)+C10_5(1,2)+C10
_5(2,1)+C10_5(2,2));

%Calculating percision performances

precision_D10_5=C10_5(2,2)/(C10_5(2,2)+C10_5(1,2));

%Calculating Averages, percision, recall's of the performance measures
sum_accuracy=0;
sum=0;
sum_accuracy=(accuracy_DecisionTree_10_1+accuracy_DecisionTree_10_2);
sum=sum_accuracy+accuracy_DecisionTree_10_3+accuracy_DecisionTree_10_4+accura
cy_DecisionTree_10_5;
Avg_accuracy=0;
Avg_accuracy=sum/5;



sum_percision=0;
Avg_percision=0;
sum_percision=(precision_D10_1+precision_D10_2+precision_D10_3+precision_D10_
4+precision_D10_5);
Avg_percision=sum_percision/5;

sum_reacll=0;
Avg_recall=0;
sum_reacll=(recall_DecisionTree_10_1+recall_DecisionTree_10_2+recall_Decision
Tree_10_3+recall_DecisionTree_10_4+recall_DecisionTree_10_5);
Avg_recall=sum_reacll/5;

%Calculating SD's of the performance measures
Accuracy_D10=[accuracy_DecisionTree_10_1 accuracy_DecisionTree_10_2
accuracy_DecisionTree_10_3 accuracy_DecisionTree_10_4
accuracy_DecisionTree_10_5];
SD_Accuracy=std(Accuracy_D10);



Percision_D10=[precision_D10_1 precision_D10_2 precision_D10_3
precision_D10_4 precision_D10_5];
SD_Percision=std(Percision_D10);

Recall_D10=[recall_DecisionTree_10_1 recall_DecisionTree_10_2
recall_DecisionTree_10_3 recall_DecisionTree_10_4 recall_DecisionTree_10_5];
SD_Recall=std(Recall_D10);


%--------------------------#3rd Question----------------------%
%Replacing the dataset with 0,1,2,3
load Data2
 sorted_data=Data2(:,1:6);
 full_data = Data2;
```

```
 A = sorted_data{:,{'pelvic_incidence'}};
 temp_array = table2array(sorted_data);
 interval_values = zeros(4,6);
 for jj=1:6
     minimum_value=min(temp_array(:,jj));
     maximum_value=max(temp_array(:,jj));
    interval_value=(maximum_value-minimum_value)/4;
    interval_values(1,jj) = interval_value+minimum_value;
    interval_values(2,jj) = 2*interval_value+minimum_value;
    interval_values(3,jj) = 3*interval_value+minimum_value;
    %interval_values(4,jj) = 4*interval_value+minimum_value;
    interval_values(4,jj) = maximum_value;
    %Replacing the values with 0's and 1's;
    for ii=1:height(sorted_data)
         if ge(temp_array(ii,jj),minimum_value) &&
le(temp_array(ii,jj),minimum_value+interval_value)
            temp_array(ii,jj) = 0;
         elseif ge(temp_array(ii,jj), minimum_value+interval_value) &&
le(temp_array(ii,jj), minimum_value+ (2*interval_value))
           temp_array(ii,jj) = 1;
        elseif ge(temp_array(ii,jj), minimum_value+ (2*interval_value)) &&
le(temp_array(ii,jj), minimum_value+ (3*interval_value))
             temp_array(ii,jj) = 2;
        else
            temp_array(ii,jj) = 3;
        end
     end
 end


 %Creating a new table with replaced values.
temp_table = array2table(temp_array, 'VariableNames', {'pelvic_incidence',
'pelvic_tilt_numeric', 'lumbar_lordosis_angle', 'sacral_slope',
'pelvic_radius', 'degree_spondylolisthesis'});
 full_data(:,1:6) = temp_table;

 %Creating 5 new traing sets with the new data

 p = randperm(size(full_data,1));%Creating a random permutation vector
 full_data_training_1 = full_data (p(1: 210), :);%Dividing the dataset2 into
210 training rows

%First DT with the new data
load full_data_training_1
FDTC_10_1=fitctree(full_data_training_1, 'class','MinLeafSize',10);%Decision
tree of 10 nodes with 1st training data

%Confusion Matrics generation with variables with 1st training data
Actual_FD_1 = full_data_training_1{1:210,{'class'}};
prediction_FD_10_1=predict(FDTC_10_1,full_data_training_1);
%Confusion Matrix for Decision Tree with 10 nodes
FD_C10_1=confusionmat(Actual_FD_1, prediction_FD_10_1);
%Calculating recall performances with seccond training data
recall_FDecisionTree_10_1=FD_C10_1(2,2)/(FD_C10_1(2,2)+FD_C10_1(2,1));
%Calculating accuracy performances
```

```
accuracy_FDecisionTree_10_1=(FD_C10_1(1,1)+FD_C10_1(2,2))/(FD_C10_1(1,1)+FD_C
10_1(1,2)+FD_C10_1(2,1)+FD_C10_1(2,2));
%Calculating percision performances
precision_FD10_1=FD_C10_1(2,2)/(FD_C10_1(2,2)+FD_C10_1(1,2));


%Second DT with the new data
p2 = randperm(size(full_data,1));%Creating a random permutation vector
full_data_training_2 = full_data (p2(1: 210), :);%Dividing the dataset2 into
210 training rows

load full_data_training_2
FDTC_10_2=fitctree(full_data_training_2, 'class','MinLeafSize',10);%Decision
tree of 10 nodes with 1st training data

%Confusion Matrics generation with variables with 1st training data
Actual_FD_2 = full_data_training_2{1:210,{'class'}};
prediction_FD_10_2=predict(FDTC_10_2,full_data_training_2);

%Confusion Matrix for Decision Tree with 10 nodes
FD_C10_2=confusionmat(Actual_FD_2, prediction_FD_10_2);

%Calculating recall performances with seccond training data
recall_FDecisionTree_10_2=FD_C10_2(2,2)/(FD_C10_2(2,2)+FD_C10_2(2,1));


%Calculating accuracy performances
accuracy_FDecisionTree_10_2=(FD_C10_2(1,1)+FD_C10_2(2,2))/(FD_C10_2(1,1)+FD_C
10_2(1,2)+FD_C10_2(2,1)+FD_C10_2(2,2));

%Calculating percision performances

precision_FD10_2=FD_C10_2(2,2)/(FD_C10_2(2,2)+FD_C10_2(1,2));


%Third DT with the new data
 p3 = randperm(size(full_data,1));%Creating a random permutation vector
 full_data_training_3 = full_data (p3(1: 210), :);%Dividing the dataset2 into
210 training rows

load full_data_training_3
FDTC_10_3=fitctree(full_data_training_3, 'class','MinLeafSize',10);%Decision
tree of 10 nodes with 1st training data

%Confusion Matrics generation with variables with 1st training data
Actual_FD_3 = full_data_training_3{1:210,{'class'}};
prediction_FD_10_3=predict(FDTC_10_3,full_data_training_3);

%Confusion Matrix for Decision Tree with 10 nodes
FD_C10_3=confusionmat(Actual_FD_3, prediction_FD_10_3);

%Calculating recall performances with seccond training data
recall_FDecisionTree_10_3=FD_C10_3(2,2)/(FD_C10_3(2,2)+FD_C10_3(2,1));
```

```
%Calculating accuracy performances
accuracy_FDecisionTree_10_3=(FD_C10_3(1,1)+FD_C10_3(2,2))/(FD_C10_3(1,1)+FD_C
10_3(1,2)+FD_C10_3(2,1)+FD_C10_3(2,2));

%Calculating percision performances
precision_FD10_3=FD_C10_3(2,2)/(FD_C10_3(2,2)+FD_C10_3(1,2));



%Fourth DT with the new data

 p4 = randperm(size(full_data,1));%Creating a random permutation vector
 full_data_training_4 = full_data (p4(1: 210), :);%Dividing the dataset2 into
210 training rows

load full_data_training_4
FDTC_10_4=fitctree(full_data_training_4, 'class','MinLeafSize',10);%Decision
tree of 10 nodes with 1st training data

%Confusion Matrics generation with variables with 1st training data
Actual_FD_4 = full_data_training_4{1:210,{'class'}};
prediction_FD_10_4=predict(FDTC_10_4,full_data_training_4);

%Confusion Matrix for Decision Tree with 10 nodes
FD_C10_4=confusionmat(Actual_FD_4, prediction_FD_10_4);

%Calculating recall performances with seccond training data
recall_FDecisionTree_10_4=FD_C10_4(2,2)/(FD_C10_4(2,2)+FD_C10_4(2,1));


%Calculating accuracy performances
accuracy_FDecisionTree_10_4=(FD_C10_4(1,1)+FD_C10_4(2,2))/(FD_C10_4(1,1)+FD_C
10_4(1,2)+FD_C10_4(2,1)+FD_C10_4(2,2));

%Calculating percision performances

precision_FD10_4=FD_C10_4(2,2)/(FD_C10_4(2,2)+FD_C10_4(1,2));


%5th DT with the new data

 p5 = randperm(size(full_data,1));%Creating a random permutation vector
full_data_training_5 = full_data (p5(1: 210), :);%Dividing the dataset2 into
210 training rows

load full_data_training_5
FDTC_10_5=fitctree(full_data_training_5, 'class','MinLeafSize',10);%Decision
tree of 10 nodes with 1st training data

%Confusion Matrics generation with variables with 1st training data
Actual_FD_5 = full_data_training_5{1:210,{'class'}};
```

```
prediction_FD_10_5=predict(FDTC_10_5,full_data_training_5);

%Confusion Matrix for Decision Tree with 10 nodes
FD_C10_5=confusionmat(Actual_FD_5, prediction_FD_10_5);

%Calculating recall performances with seccond training data
recall_FDecisionTree_10_5=FD_C10_5(1,1)/(FD_C10_5(2,2)+FD_C10_5(2,1));


%Calculating accuracy performances
accuracy_FDecisionTree_10_5=(FD_C10_5(1,1)+FD_C10_5(2,2))/(FD_C10_5(1,1)+FD_C
10_5(1,2)+FD_C10_5(2,1)+FD_C10_5(2,2));

%Calculating percision performances

precision_FD10_5=FD_C10_5(2,2)/(FD_C10_5(2,2)+FD_C10_5(1,2));

%Calculating the Accuracy, percision, recall of the new data set

sum_accuracy_FD=0;
sum_FD=0;
sum_accuracy_FD=(accuracy_FDecisionTree_10_1+accuracy_FDecisionTree_10_2);
sum_FD=sum_accuracy_FD+accuracy_FDecisionTree_10_3+accuracy_FDecisionTree_10_
4+accuracy_FDecisionTree_10_5;
Avg_accuracy_FD=0;
Avg_accuracy_FD=sum_FD/5;

percision_FD=0;
Avg_percision_FD=0;
sum_percision_FD=(precision_FD10_1+precision_FD10_2+precision_FD10_3+precisio
n_FD10_4+precision_FD10_5);
Avg_percision_FD=sum_percision_FD/5;

sum_reacll_FD=0;
Avg_recall_FD=0;
sum_reacll_FD=(recall_FDecisionTree_10_1+recall_FDecisionTree_10_2+recall_FDe
cisionTree_10_3+recall_FDecisionTree_10_4+recall_FDecisionTree_10_5);

Avg_recall_FD=sum_reacll_FD/5;

%Calculating SD's of the performance measures
Accuracy_FD10=[accuracy_FDecisionTree_10_1 accuracy_FDecisionTree_10_2
accuracy_FDecisionTree_10_3 accuracy_FDecisionTree_10_4
accuracy_FDecisionTree_10_5];
SD_FD_accuracy=std(Accuracy_FD10)

Percision_FD10=[precision_FD10_1 precision_FD10_2 precision_FD10_3
precision_FD10_4 precision_FD10_5];
SD_FD_Percision=std(Percision_FD10)

Recall_FD10=[recall_FDecisionTree_10_1 recall_FDecisionTree_10_2
recall_FDecisionTree_10_3 recall_FDecisionTree_10_4
recall_FDecisionTree_10_5];
SD_FD_Recall=std(Recall_FD10)
```

```
%-------------------------2nd Question------------------------------------%
```

Import the dataset using the import option in matlab.

```
r = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training = Data3 (r(1: 210), :);%Dividing the dataset2 into 210
training rows
%Data2_test = Data2 (k(211: end), :);%Dividing the dataset2 into 100 test
rows


load Data3_training
D3TC_5=fitctree(Data3_training, 'class','MinLeafSize',5);
%Decision Trees Constructing
D3TC_10=fitctree(Data3_training, 'class','MinLeafSize',10);
D3TC_15=fitctree(Data3_training, 'class','MinLeafSize',15);
D3TC_20=fitctree(Data3_training, 'class','MinLeafSize',20);
D3TC_25=fitctree(Data3_training, 'class','MinLeafSize',25);

view((D3TC_25),'mode','graph')

%---------------------------------Q1.2b-------------------------%
%Confusion Matrics
Actual_D3 = Data3_training{1:210,{'class'}};
%Confusion Matrix for Decision Tree with 5 nodes
prediction_D3_5=predict(D3TC_5,Data3_training);
C5_D3=confusionmat(Actual_D3, prediction_D3_5);

%Confusion Matrix for Decision Tree with 10 nodes
prediction__D3_10=predict(D3TC_10,Data3_training);
C10_D3=confusionmat(Actual_D3, prediction__D3_10);

%Confusion Matrix for Decision Tree with 15 nodes
prediction__D3_15=predict(D3TC_15,Data3_training);
C15_D3=confusionmat(Actual_D3, prediction__D3_15);

%Confusion Matrix for Decision Tree with 20 nodes
prediction__D3_20=predict(D3TC_20,Data3_training);
C20_D3=confusionmat(Actual_D3, prediction__D3_20);
%[c_matrixp,Result]= confusion.getMatrix(Actual_D3,prediction_D3_20);


%Confusion Matrix for Decision Tree with 25 nodes
prediction_D3_25=predict(D3TC_25,Data3_training);
C25_D3=confusionmat(Actual_D3, prediction_D3_25);

%Calculating recall performances

recall_Decision3Tree_5_Normal=C5_D3(2,2)/(C5_D3(2,2)+C5_D3(2,1)+C5_D3(2,3));
recall_Decision3Tree_10_Normal=C10_D3(2,2)/(C10_D3(2,2)+C10_D3(2,1)+C10_D3(2,
3));
```

```
recall_Decision3Tree_15_Normal=C15_D3(2,2)/(C15_D3(2,2)+C15_D3(2,1)+C15_D3(2,3));
recall_Decision3Tree_20_Normal=C20_D3(2,2)/(C20_D3(2,2)+C20_D3(2,1)+C20_D3(2,3));
recall_Decision3Tree_25_Normal=C25_D3(2,2)/(C25_D3(2,2)+C25_D3(2,1)+C25_D3(2,3));


%Calculating accuracy performances
accuracy_DecisionTree_5=(C5_D3(1,1)+C5_D3(2,2)+C5_D3(3,3))/(C5_D3(1,1)+C5_D3(1,2)+C5_D3(2,1)+C5_D3(2,2)+C5_D3(3,3)+C5_D3(3,1)+C5_D3(3,2)+C5_D3(1,3)+C5_D3(2,3));
accuracy_DecisionTree_10=(C10_D3(1,1)+C10_D3(2,2)+C10_D3(3,3))/(C10_D3(1,1)+C10_D3(1,2)+C10_D3(2,1)+C10_D3(2,2)+C10_D3(3,3)+C10_D3(3,1)+C10_D3(3,2)+C10_D3(1,3)+C10_D3(2,3));
accuracy_DecisionTree_15=(C15_D3(1,1)+C15_D3(2,2)+C15_D3(3,3))/(C15_D3(1,1)+C15_D3(1,2)+C15_D3(2,1)+C15_D3(2,2)+C15_D3(3,3)+C15_D3(3,1)+C15_D3(3,2)+C15_D3(1,3)+C15_D3(2,3));
accuracy_DecisionTree_20=(C20_D3(1,1)+C20_D3(2,2)+C20_D3(3,3))/(C20_D3(1,1)+C20_D3(1,2)+C20_D3(2,1)+C20_D3(2,2)+C20_D3(3,3)+C20_D3(3,1)+C20_D3(3,2)+C20_D3(1,3)+C20_D3(2,3));
accuracy_DecisionTree_25=(C25_D3(1,1)+C25_D3(2,2)+C25_D3(3,3))/(C25_D3(1,1)+C25_D3(1,2)+C25_D3(2,1)+C25_D3(2,2)+C25_D3(3,3)+C25_D3(3,1)+C25_D3(3,2)+C25_D3(1,3)+C25_D3(2,3));

%Calculating percision performances

precision_D5_normal=C5_D3(2,2)/(C5_D3(2,2)+C5_D3(1,2)+C5_D3(3,2));
precision_D10_normal=C10_D3(2,2)/(C10_D3(2,2)+C10_D3(1,2)+C10_D3(3,2));
precision_D15_normal=C15_D3(2,2)/(C15_D3(2,2)+C15_D3(1,2)+C15_D3(3,2));
precision_D20_normal=C20_D3(2,2)/(C20_D3(2,2)+C20_D3(1,2)+C20_D3(3,2));
precision_D25_normal=C25_D3(2,2)/(C25_D3(2,2)+C25_D3(1,2)+C25_D3(3,2));



plot_accuracy=[accuracy_DecisionTree_5 accuracy_DecisionTree_10 accuracy_DecisionTree_15 accuracy_DecisionTree_20 accuracy_DecisionTree_25];
plot_recall=[recall_Decision3Tree_5_Normal recall_Decision3Tree_10_Normal recall_Decision3Tree_15_Normal recall_Decision3Tree_20_Normal recall_Decision3Tree_25_Normal];
plot_percision=[precision_D5_normal precision_D10_normal precision_D15_normal precision_D20_normal precision_D25_normal];

figure;
bar(plot_accuracy, 0.5);
ylabel('Accuracy', 'FontSize', 15);
title('Accuracy plot');
names = {'Decision tree 1', 'Decision tree 2', 'Decision tree 3', 'Decision tree 4', 'Decision tree 5'};
set(gca, 'xticklabel', names, 'FontSize', 15);


figure;
bar(plot_recall, 0.5);
ylabel('Recall', 'FontSize', 15);
```

```
title('Recall plot');
names = {'Decision tree 1', 'Decision tree 2', 'Decision tree 3', 'Decision
tree 4', 'Decision tree 5'};
set(gca, 'xticklabel', names, 'FontSize', 15);


figure;
bar(plot_percision, 0.5);
ylabel('Percision', 'FontSize', 15);
title(' Percesion plot');
names = {'Decision tree 1', 'Decision tree 2', 'Decision tree 3', 'Decision
tree 4', 'Decision tree 5'};
set(gca, 'xticklabel', names, 'FontSize', 15);



%Generating 5 datasets

r1 = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training_1 = Data3 (r1(1: 210), :);%Dividing the dataset2 into 210
training rows

load Data3_training_1
D3TC_10_1=fitctree(Data3_training_1, 'class','MinLeafSize',10);

Actual_D3_1 = Data3_training_1{1:210,{'class'}};
prediction__D3_10_1=predict(D3TC_10_1,Data3_training_1);
C10_D3_1=confusionmat(Actual_D3_1, prediction__D3_10_1);

recall_Decision3Tree_10_Normal_1=C10_D3_1(2,2)/(C10_D3_1(2,2)+C10_D3_1(2,1)+C
10_D3_1(2,3));

accuracy_DecisionTree_10_1=(C10_D3_1(1,1)+C10_D3_1(2,2)+C10_D3_1(3,3))/(C10_D
3_1(1,1)+C10_D3_1(1,2)+C10_D3_1(2,1)+C10_D3_1(2,2)+C10_D3_1(3,3)+C10_D3_1(3,1
)+C10_D3_1(3,2)+C10_D3_1(1,3)+C10_D3_1(2,3));

precision_D10_normal_1=C10_D3_1(2,2)/(C10_D3_1(2,2)+C10_D3_1(1,2)+C10_D3_1(3,
2));




r2 = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training_2 = Data3 (r2(1: 210), :);%Dividing the dataset2 into 210
training rows

load Data3_training_2
D3TC_10_2=fitctree(Data3_training_2, 'class','MinLeafSize',10);

Actual_D3_2 = Data3_training_2{1:210,{'class'}};
prediction__D3_10_2=predict(D3TC_10_2,Data3_training_2);
C10_D3_2=confusionmat(Actual_D3_2, prediction__D3_10_2);
```

```
recall_Decision3Tree_10_Normal_2=C10_D3_2(2,2)/(C10_D3_2(2,2)+C10_D3_2(2,1)+C
10_D3_2(2,3));

accuracy_DecisionTree_10_2=(C10_D3_2(1,1)+C10_D3_2(2,2)+C10_D3_2(3,3))/(C10_D
3_2(1,1)+C10_D3_2(1,2)+C10_D3_2(2,1)+C10_D3_2(2,2)+C10_D3_2(3,3)+C10_D3_2(3,1
)+C10_D3_2(3,2)+C10_D3_2(1,3)+C10_D3_2(2,3));

precision_D10_normal_2=C10_D3_2(2,2)/(C10_D3_2(2,2)+C10_D3_2(1,2)+C10_D3_2(3,
2));



r3 = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training_3 = Data3 (r3(1: 210), :);%Dividing the dataset2 into 210
training rows

load Data3_training_3
D3TC_10_3=fitctree(Data3_training_3, 'class','MinLeafSize',10);

Actual_D3_3 = Data3_training_3{1:210,{'class'}};
prediction__D3_10_3=predict(D3TC_10_3,Data3_training_3);
C10_D3_3=confusionmat(Actual_D3_3, prediction__D3_10_3);

recall_Decision3Tree_10_Normal_3=C10_D3_3(2,2)/(C10_D3_3(2,2)+C10_D3_3(2,1)+C
10_D3_3(2,3));

accuracy_DecisionTree_10_3=(C10_D3_3(1,1)+C10_D3_3(2,2)+C10_D3_3(3,3))/(C10_D
3_3(1,1)+C10_D3_3(1,2)+C10_D3_3(2,1)+C10_D3_3(2,2)+C10_D3_3(3,3)+C10_D3_3(3,1
)+C10_D3_3(3,2)+C10_D3_3(1,3)+C10_D3_3(2,3));

precision_D10_normal_3=C10_D3_3(2,2)/(C10_D3_3(2,2)+C10_D3_3(1,2)+C10_D3_3(3,
2));



r4 = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training_4 = Data3 (r4(1: 210), :);%Dividing the dataset2 into 210
training rows

load Data3_training_4
D3TC_10_4=fitctree(Data3_training_4, 'class','MinLeafSize',10);

Actual_D3_4 = Data3_training_4{1:210,{'class'}};
prediction__D3_10_4=predict(D3TC_10_4,Data3_training_4);
C10_D3_4=confusionmat(Actual_D3_4, prediction__D3_10_4);

recall_Decision3Tree_10_Normal_4=C10_D3_4(2,2)/(C10_D3_4(2,2)+C10_D3_4(2,1)+C
10_D3_4(2,3));

accuracy_DecisionTree_10_4=(C10_D3_4(1,1)+C10_D3_4(2,2)+C10_D3_4(3,3))/(C10_D
3_4(1,1)+C10_D3_4(1,2)+C10_D3_4(2,1)+C10_D3_4(2,2)+C10_D3_4(3,3)+C10_D3_4(3,1
)+C10_D3_4(3,2)+C10_D3_4(1,3)+C10_D3_4(2,3));
```

```
precision_D10_normal_4=C10_D3_4(2,2)/(C10_D3_4(2,2)+C10_D3_4(1,2)+C10_D3_4(3,
2));




r5 = randperm(size(Data3,1));%Creating a random permutation vector
Data3_training_5 = Data3 (r5(1: 210), :);%Dividing the dataset2 into 210
training rows

load Data3_training_5
D3TC_10_5=fitctree(Data3_training_5, 'class','MinLeafSize',10);

Actual_D3_5 = Data3_training_5{1:210,{'class'}};
prediction__D3_10_5=predict(D3TC_10_5,Data3_training_5);
C10_D3_5=confusionmat(Actual_D3_5, prediction__D3_10_5);

recall_Decision3Tree_10_Normal_5=C10_D3_5(2,2)/(C10_D3_5(2,2)+C10_D3_5(2,1)+C
10_D3_5(2,3));

accuracy_DecisionTree_10_5=(C10_D3_5(1,1)+C10_D3_5(2,2)+C10_D3_5(3,3))/(C10_D
3_5(1,1)+C10_D3_5(1,2)+C10_D3_5(2,1)+C10_D3_5(2,2)+C10_D3_5(3,3)+C10_D3_5(3,1
)+C10_D3_5(3,2)+C10_D3_5(1,3)+C10_D3_5(2,3));

precision_D10_normal_5=C10_D3_5(2,2)/(C10_D3_5(2,2)+C10_D3_5(1,2)+C10_D3_5(3,
2));


sum_accuracy_D5=accuracy_DecisionTree_10_1+accuracy_DecisionTree_10_2+accurac
y_DecisionTree_10_3+accuracy_DecisionTree_10_4+accuracy_DecisionTree_10_5;
Avg_accuracy_D5=sum_accuracy_D5/5;

sum_percision_D5=precision_D10_normal_1+precision_D10_normal_2+precision_D10_
normal_3+precision_D10_normal_4+precision_D10_normal_5;
Avg_percision_D5=sum_percision_D5/5;

sum_recall_D5=recall_Decision3Tree_10_Normal_1+recall_Decision3Tree_10_Normal
_2+recall_Decision3Tree_10_Normal_3+recall_Decision3Tree_10_Normal_4+recall_D
ecision3Tree_10_Normal_5;
Avg_recall_D5=sum_recall_D5/5;


Accuracy_D5=[accuracy_DecisionTree_10_1 accuracy_DecisionTree_10_2
accuracy_DecisionTree_10_3 accuracy_DecisionTree_10_4
accuracy_DecisionTree_10_5];
percision_D5=[precision_D10_normal_1 precision_D10_normal_2
precision_D10_normal_3 precision_D10_normal_4 precision_D10_normal_5];
recall_D5=[recall_Decision3Tree_10_Normal_1 recall_Decision3Tree_10_Normal_2
recall_Decision3Tree_10_Normal_3 recall_Decision3Tree_10_Normal_4
recall_Decision3Tree_10_Normal_5];

SD_Accuracy_D5=std(Accuracy_D5);
SD_Recall_D5=std(recall_D5);
SD_percision=std(percision_D5);
```