

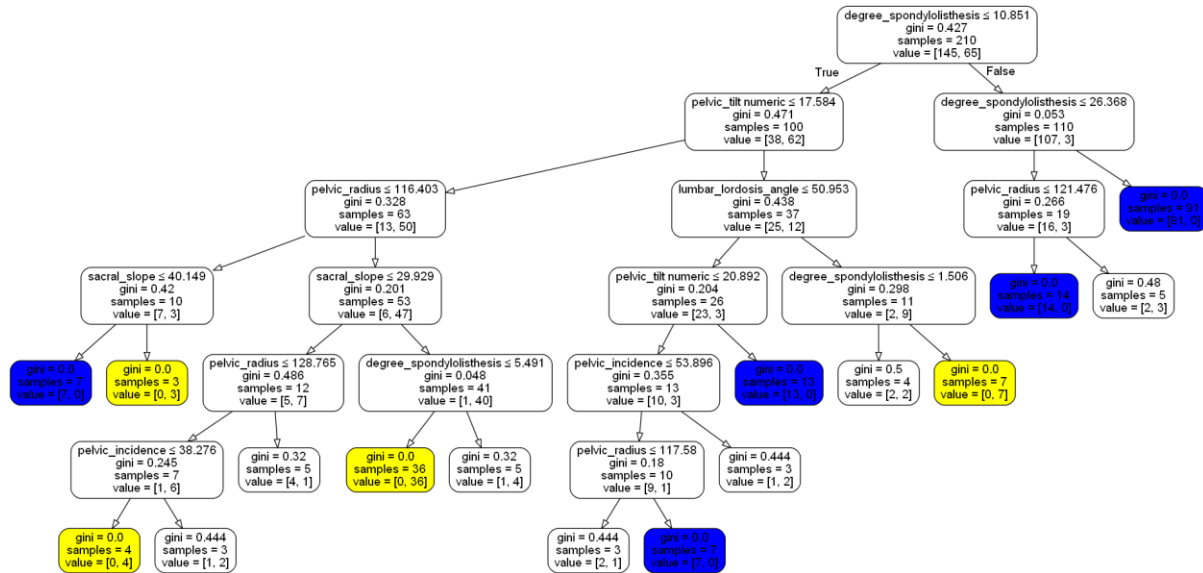
Intelligent Data Analysis – Assignment_1

Name: Piyush Sanghi MID: M12952017

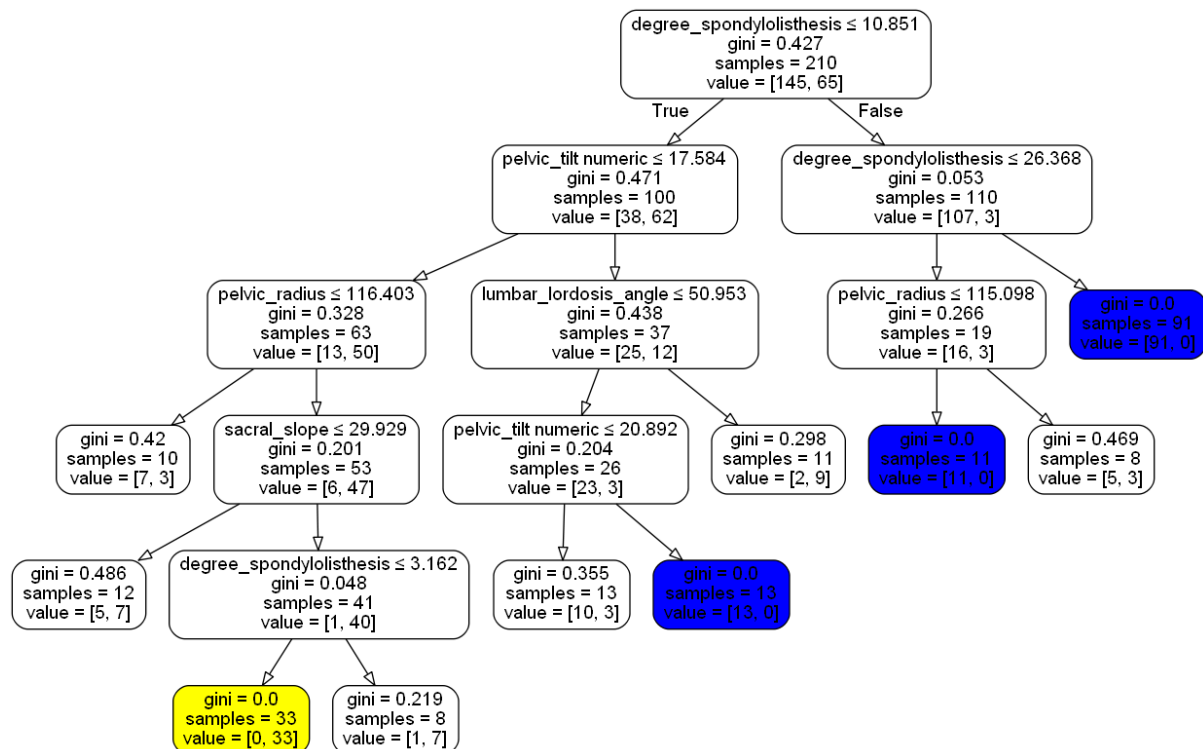
Programming language used: Python 3 (Code, explanation and output screenshots at end of the file)

Answer 1a:

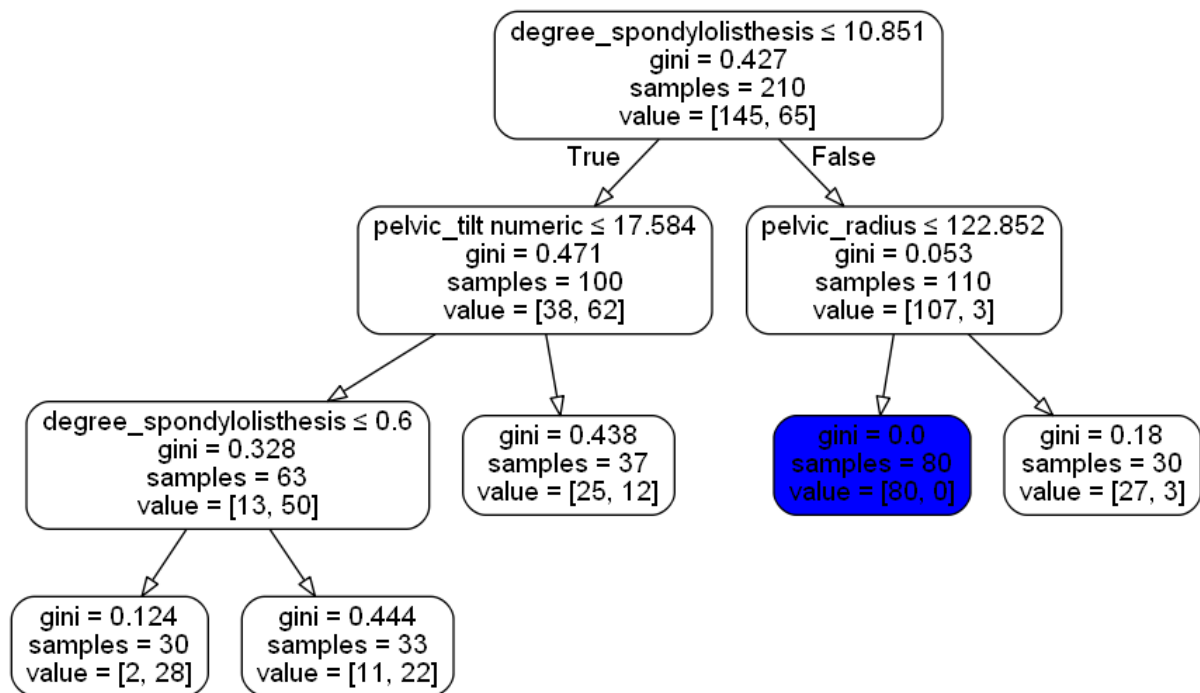
Decision Tree classifier for Minimum 3 records at leaf nodes



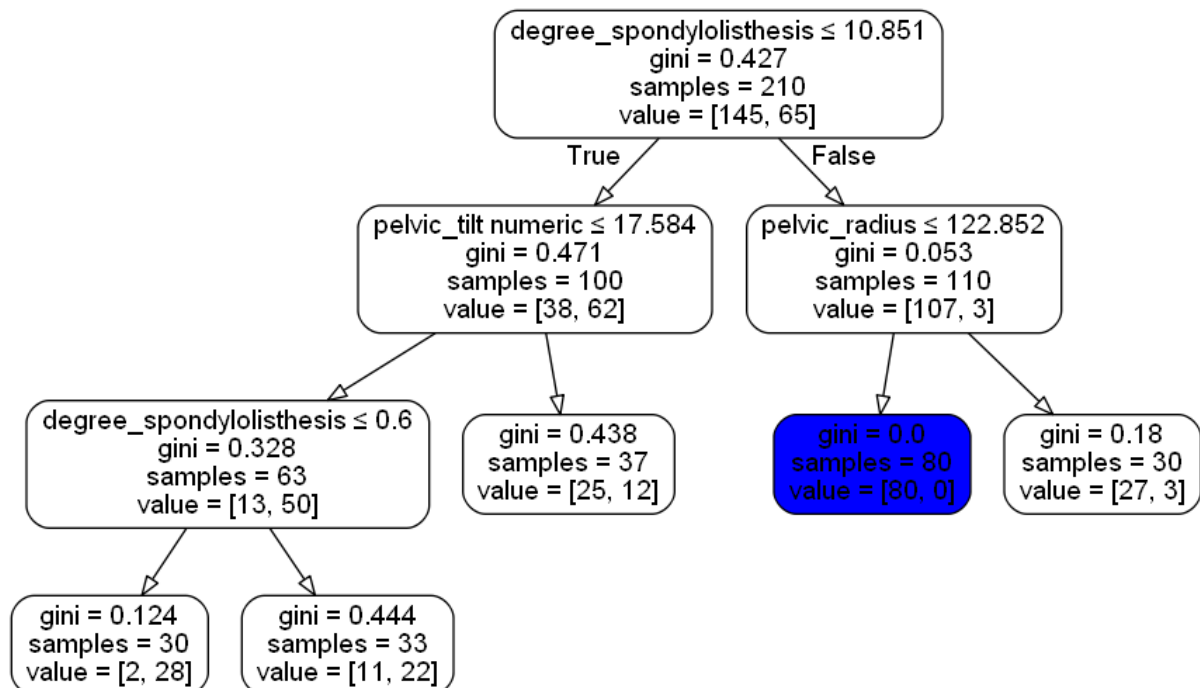
Decision Tree classifier for minimum 8 records at leaf nodes



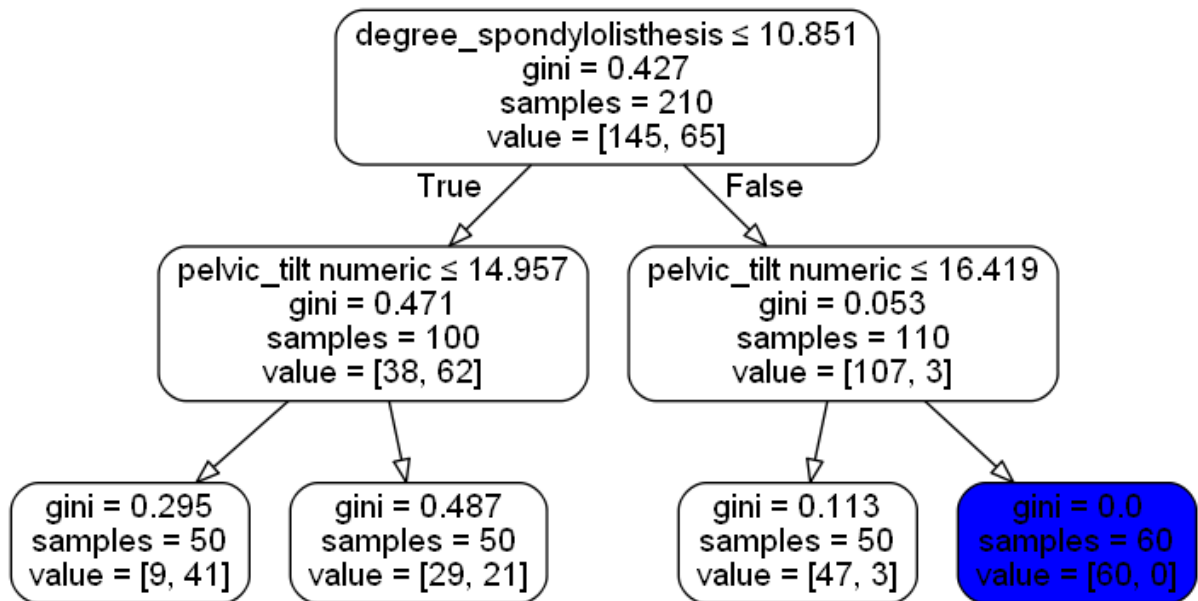
Decision Tree classifier for minimum 12 records at leaf nodes



Decision Tree classifier for minimum 30 records at leaf nodes



Decision Tree classifier for minimum 50 records at leaf nodes



We notice that all the trees root attribute is the same, but as the number of minimum leaf nodes are decreasing we can notice that number of attributes utilized are increasing.

From the above Decision trees we can infer that trees with minimum samples at nodes as 50 and 30 are not really giving any much useful information as it actually undergoes under-fitting.

And the tree with minimum 3 samples at nodes is very complex and undergoes overfitting with many pure singleton classes.

Decision trees with 8 and 12 minimum number of nodes are having good classification but it makes it little difficult to choose between the two as the one with 8 minimum nodes has more pure classes but may sometimes lead to over classification and the one with 12 samples at nodes can sometimes be under classified.

We would like to use **Occam's razor** rule and choose a tree which is not very complex and uses attributes which are the best fits, hence by observation of different given trees, tree with **12** minimum number of samples at leaf nodes look a **better choice** than the remaining as it **not complex** and it **neglects noise** and hence **avoids overfitting**.

Answer 1b:

Accuracy, Precision, Recall for Decision tree with minimum 3 data sets at leaf node are

0.77, 0.7473958333333333, 0.7505494505494505

Accuracy, Precision, Recall for Decision tree with minimum 8 data sets at leaf node are

0.78, 0.7606358111266948, 0.7450549450549451

Accuracy, Precision, Recall for Decision tree with minimum 12 data sets at leaf node are

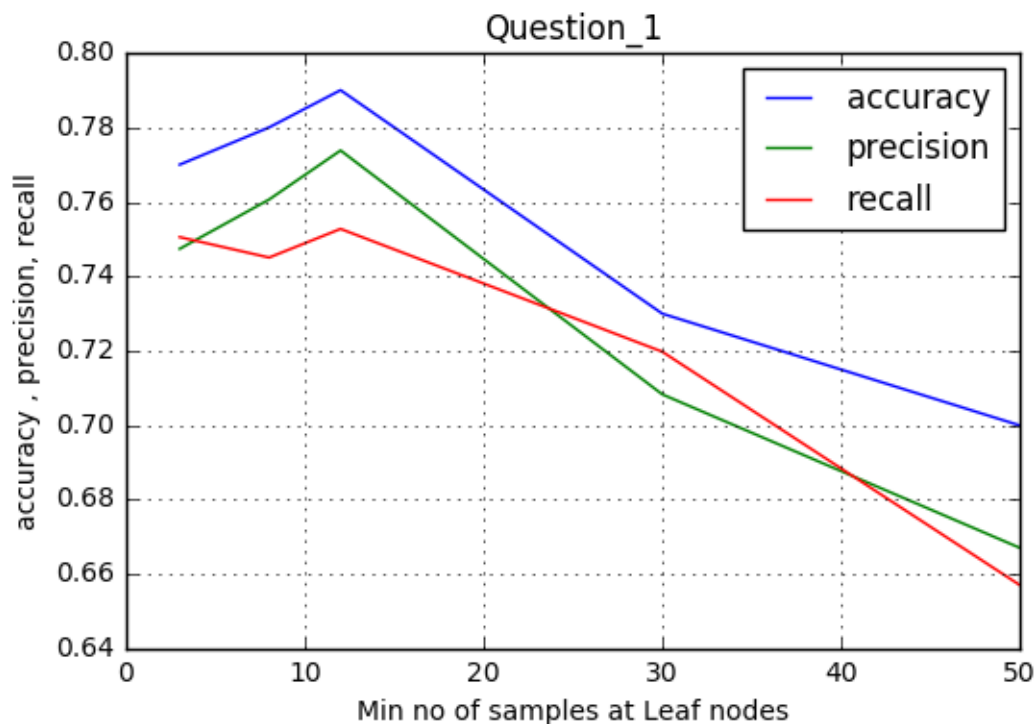
0.79, 0.7738095238095237, 0.7527472527472527

Accuracy, Precision, Recall for Decision tree with minimum 30 data sets at leaf node are

0.73, 0.7083333333333333, 0.7197802197802198

Accuracy, Precision, Recall for Decision tree with minimum 50 data sets at leaf node are

0.7, 0.6671341748480599, 0.6571428571428571

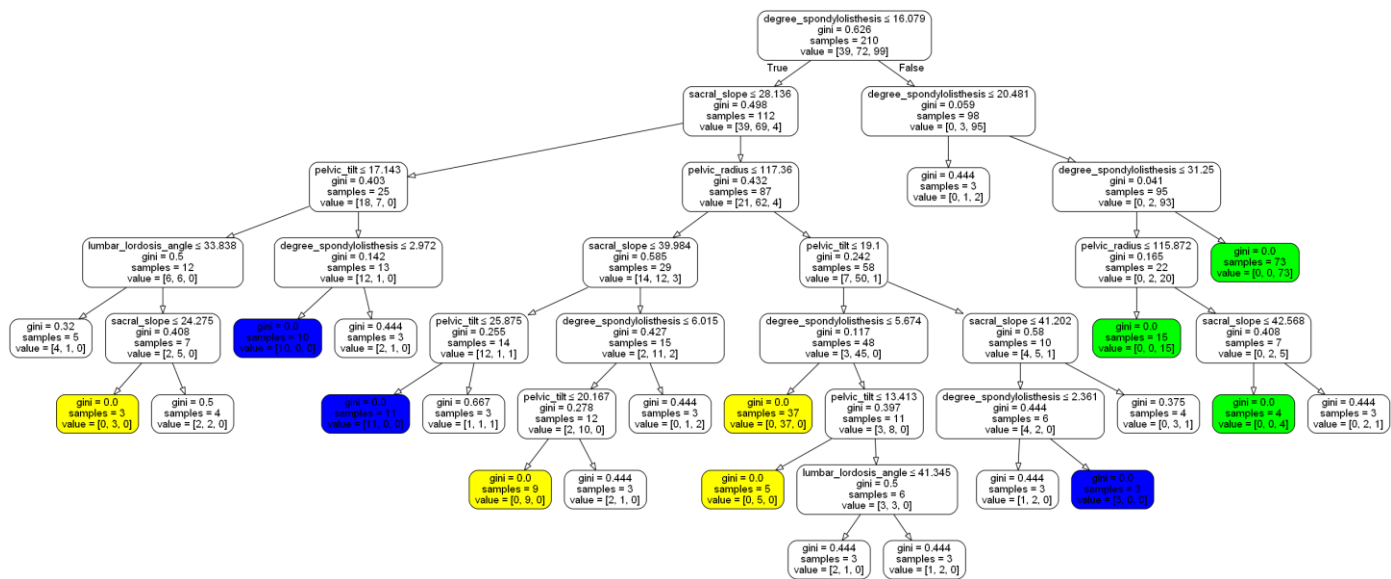


Hence by looking at the graph it is very clear that **Decision tree with minimum 12 samples at leaf nodes has the best Precision, Recall and Accuracy**. Hence it gives the best classification when compared to other trees.

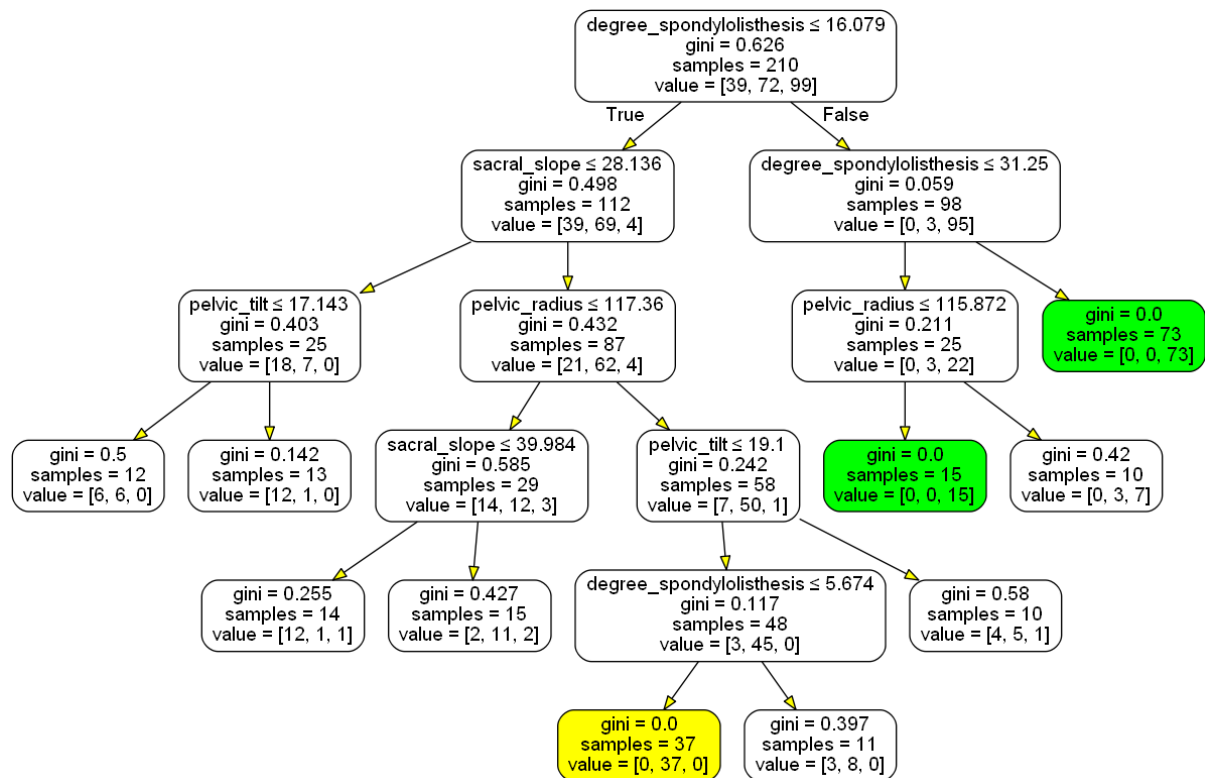
We can even notice a trend i.e. initially with increase in number of minimum samples at leaf nodes, **accuracy and precision increases till 12 minimum samples at leaf nodes and then it gradually decreases**. By this we can derive that initially we were overclassifying the data, hence there is less accuracy and then after a point we are under classifying the data. **This shows that we will get a point in the decision tree classification which can be the best fit for minimum number of leaf nodes by seeing the plot.**

Answer 2a:

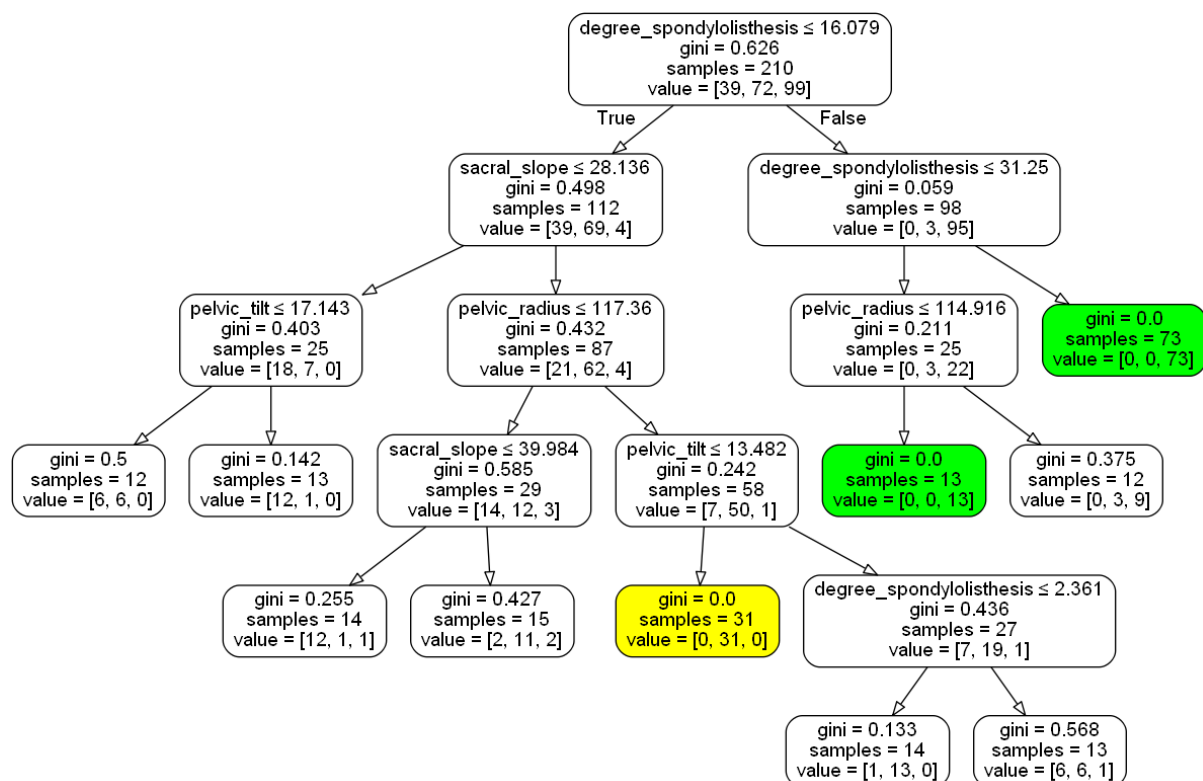
Decision Tree classifier for minimum 3 records at leaf nodes



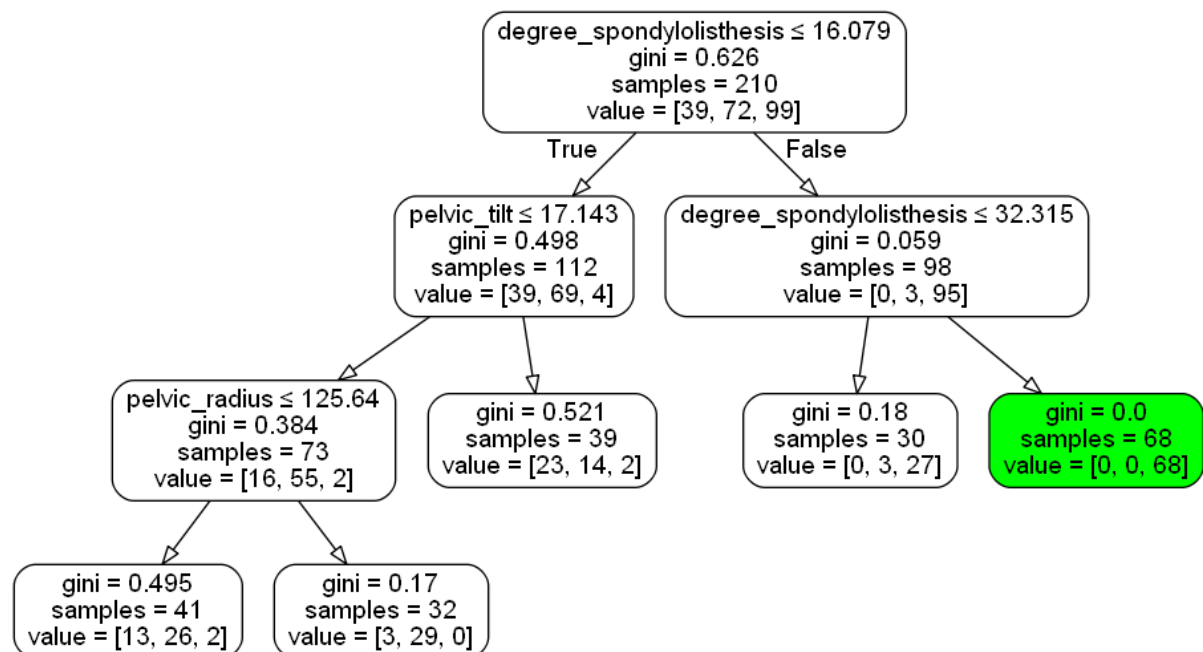
Decision Tree classifier for minimum 8 records at leaf nodes



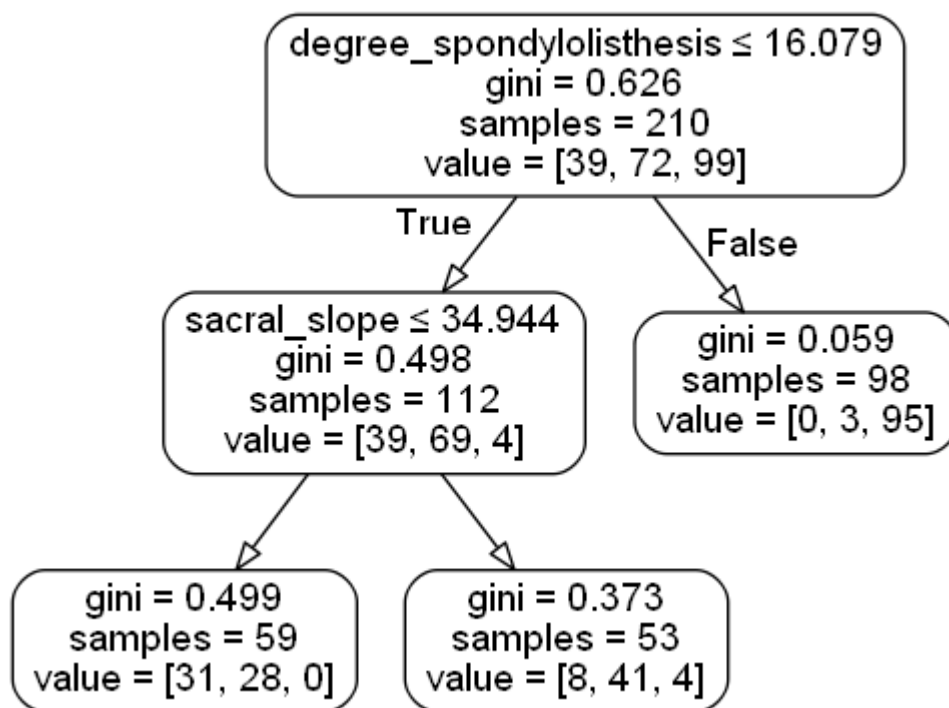
Decision Tree classifier for minimum 12 records at leaf nodes



Decision Tree classifier for minimum 30 records at leaf nodes



Decision Tree classifier for minimum 50 records at leaf nodes



We notice that in this case for 3 minimum samples at leaf node, it is same like the first question and is too complex and over-fitted.

For the decision tree with 8 minimum samples at leaf nodes, when compared to the tree with 12 minimum samples at leaf nodes is less complex and even has equal number of pure nodes , **hence when compared with 12 minimum samples at nodes the decision tree with 8 minimum samples at leaf nodes is better** .

Trees with 50 and 30 nodes are again under classified and have no much significance as the first one.

The tree with 50 nodes doesn't even have one node with good classification.

Hence of all the above we can choose the decision tree with 12 minimum samples at leaf nodes.

Answer 2b:

Accuracy for Decision Tree with 3 data sets at leaf node is 0.85

Precision and recall for class **Normal** is 0.8, 0.9411764705882353

Precision and recall for class **Spondylolisthesis** is 0.7142857142857143, 0.9411764705882353

Precision and recall for class **Hernia** is 0.7547169811320756, 0.9411764705882353

Accuracy for Decision Tree with 8 data sets at leaf node are 0.86

Precision, recall for class Normal is 0.7333333333333333, 1.0

Precision, recall for class Spondylolisthesis is 0.7857142857142857, 0.9803921568627451

Precision, recall for class Hernia is 0.7586206896551724, 0.99009900990099

Accuracy for Decision Tree with 12 data sets at leaf node are 0.82

Precision, recall for class Normal is 0.7083333333333334, 1.0

Precision, recall for class Spondylolisthesis is 0.6071428571428571, 0.9803921568627451

Precision, recall for class Hernia is 0.6538461538461539, 0.99009900990099

Accuracy for Decision Tree with 30 data sets at leaf node are 0.77

Precision, recall for class Normal is 0.5882352941176471, 1.0

Precision, recall for class Spondylolisthesis is 0.7142857142857143, 0.9803921568627451

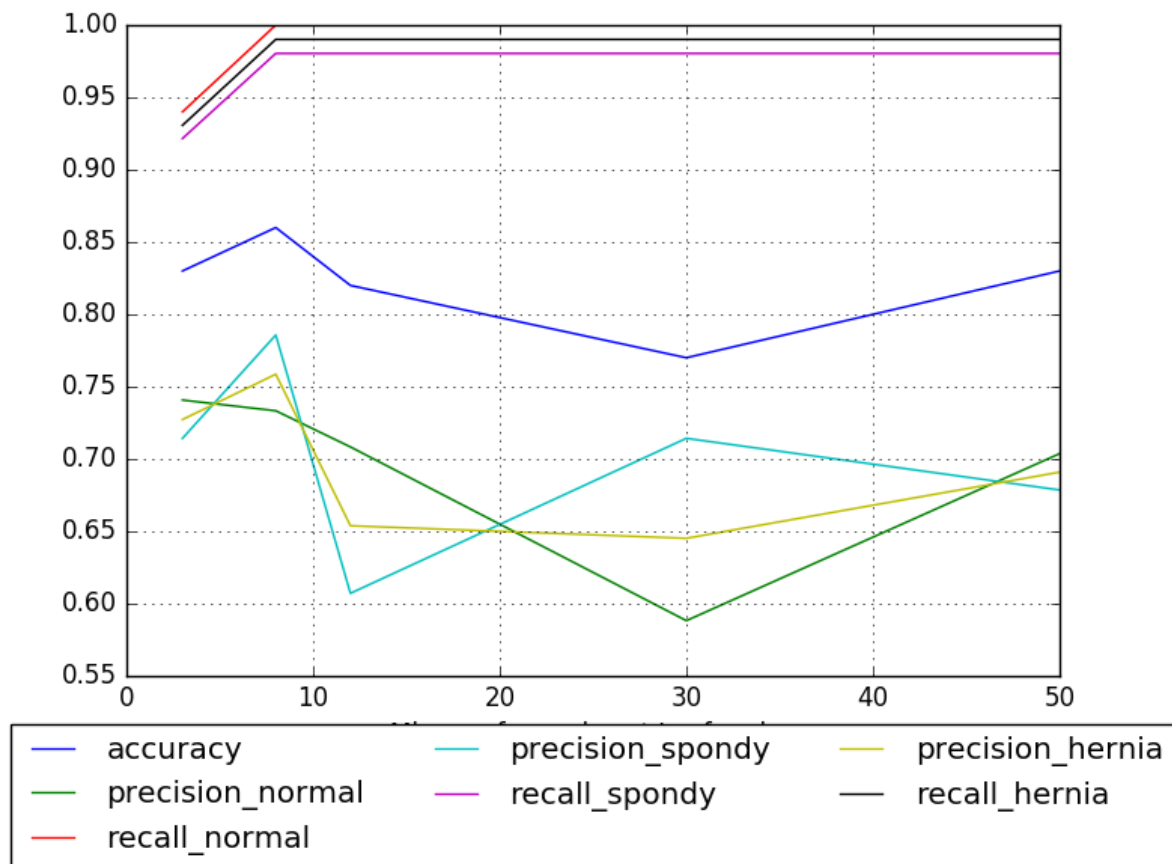
Precision, recall for class hernia is 0.6451612903225806, 0.99009900990099

Accuracy for Decision Tree with 50 data sets at leaf node are 0.83

Precision, recall for class Normal is 0.7037037037037037, 1.0

Precision, recall for class Spondylolisthesis is 0.6785714285714286, 0.9803921568627451

Precision, recall for class Hernia is 0.6909090909090909, 0.99009900990099



Answer 2c:

For 3 minimum samples at leaf node it is same like the first question and is too complex and over-fitted.

When compared to the first question this tree with 8 minimum samples at leaf nodes has less complexity and better classification than the one in first question.

Here the tree with 12 minimum samples at leaf nodes has increased complexity than the one in first question.

Decision trees with 30, 50 minimum samples at leaf nodes are comparable to the one with the first question as they are under classified and do not really provide meaningful information or conclusions.

Conclusion and comparison seeing the Graph:

We can notice that Precision, Recall and Accuracy values of most of the classes are higher at 8 minimum samples at leaf nodes. Hence it can be inferred that it is the best classification.

The general trend is that most of the values are increasing up-to 8 minimum samples at leaf nodes and then they decrease and get random depending on the data.

Answer 3a:

Boundaries for **pelvic_incidence** are 26.14792141, 52.06945121, 77.99098101, 103.9125108, 129.8340406

Boundaries for **pelvic_tilt** numeric are -6.55494835, 7.44175464, 21.43845763, 35.43516061, 49.4318636

Boundaries for **lumbar_lordosis** angle are 14.0, 41.93559638, 69.87119275, 97.80678912, 125.7423855

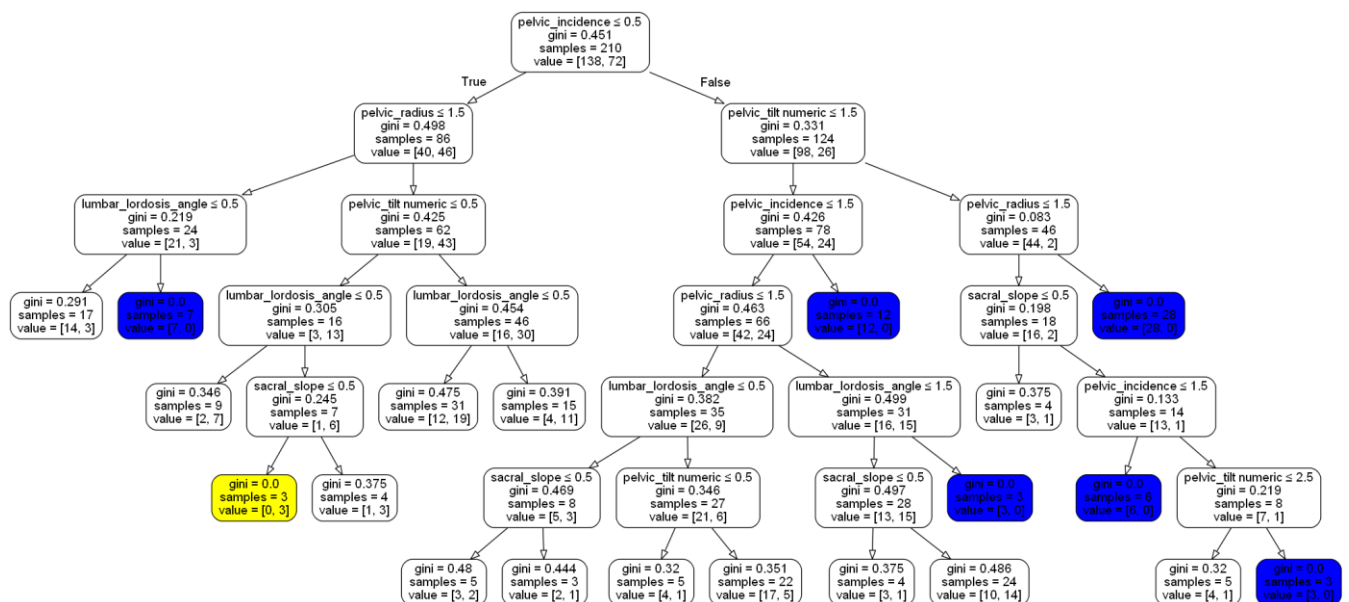
Boundaries for **sacral_slope** are 13.3669307, 40.38258942, 67.39824815, 94.41390687, 121.4295656

Boundaries for **pelvic_radius** are 70.08257486, 93.32969127, 116.57680768, 139.82392409, 163.0710405

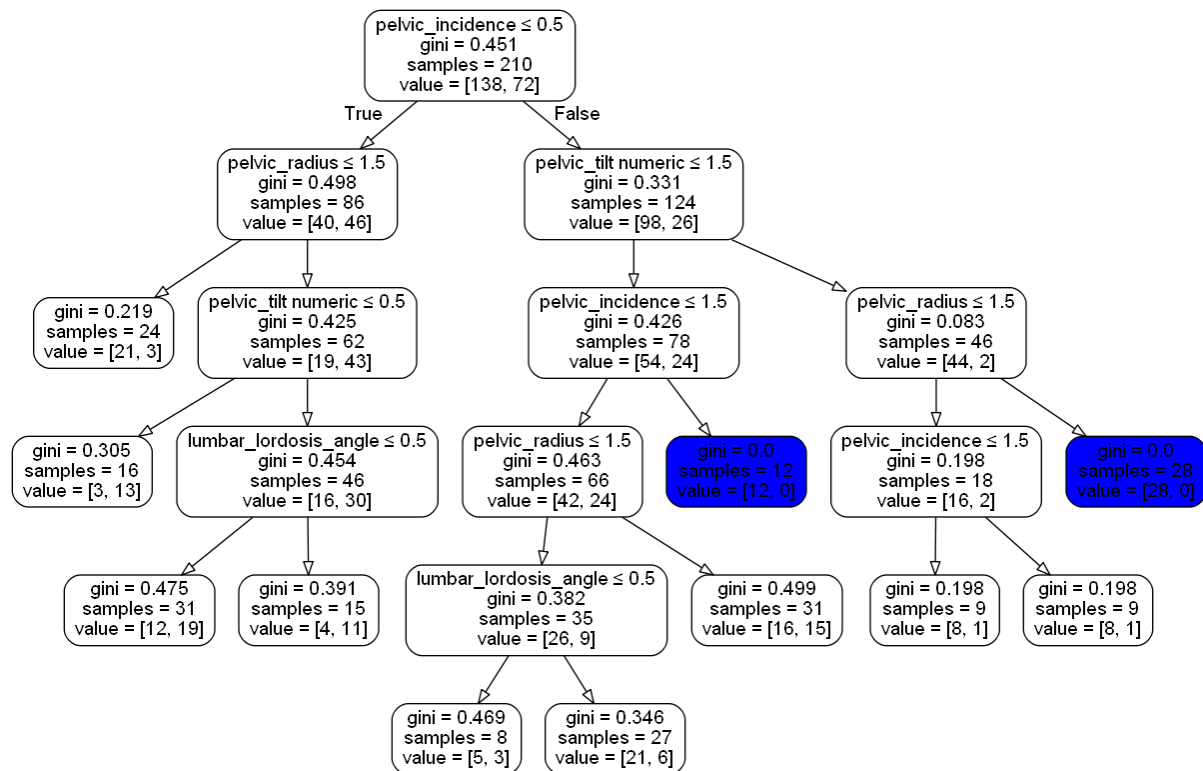
Boundaries for **degree_spondylolisthesis** are -11.05817866, 96.34213653, 203.74245172, 311.14276691, 418.5430821

Answer 3b:

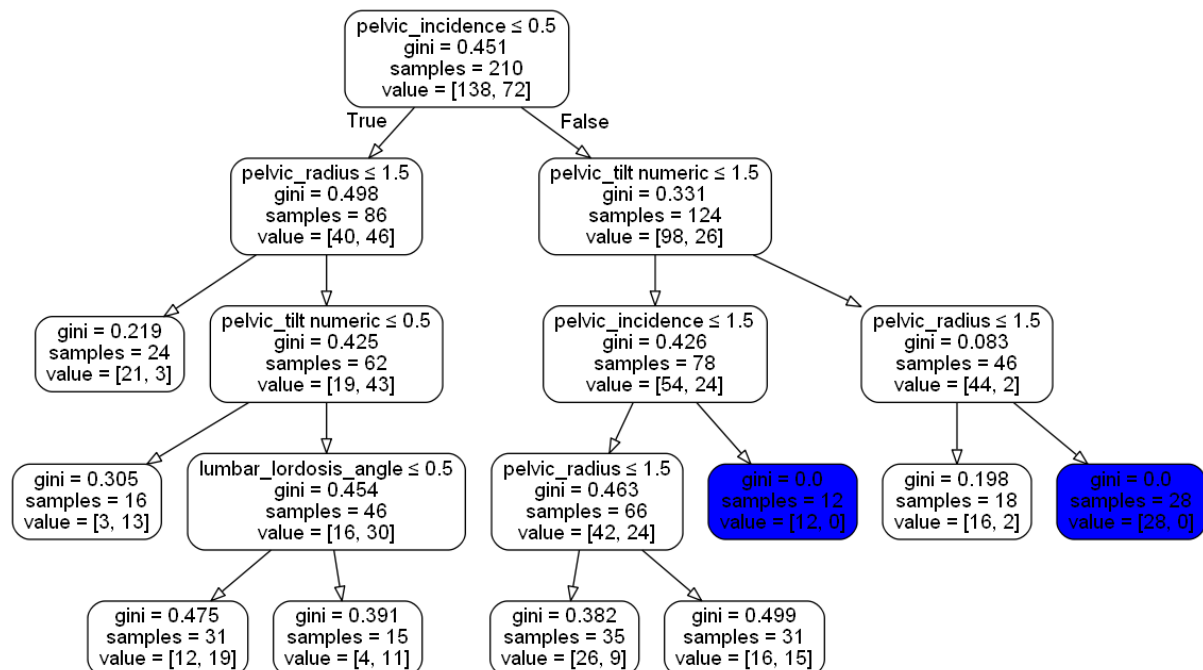
Decision Tree classifier for Minimum 3 records at leaf nodes



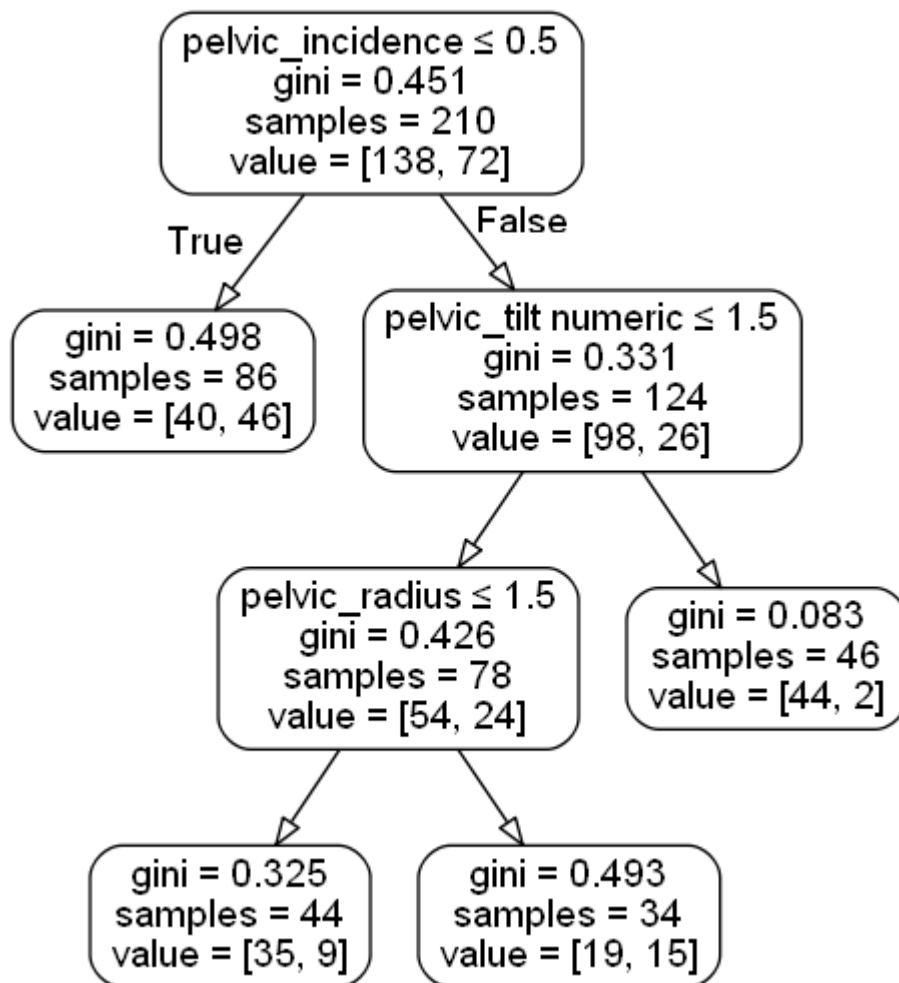
Decision Tree classifier for Minimum 8 records at leaf nodes



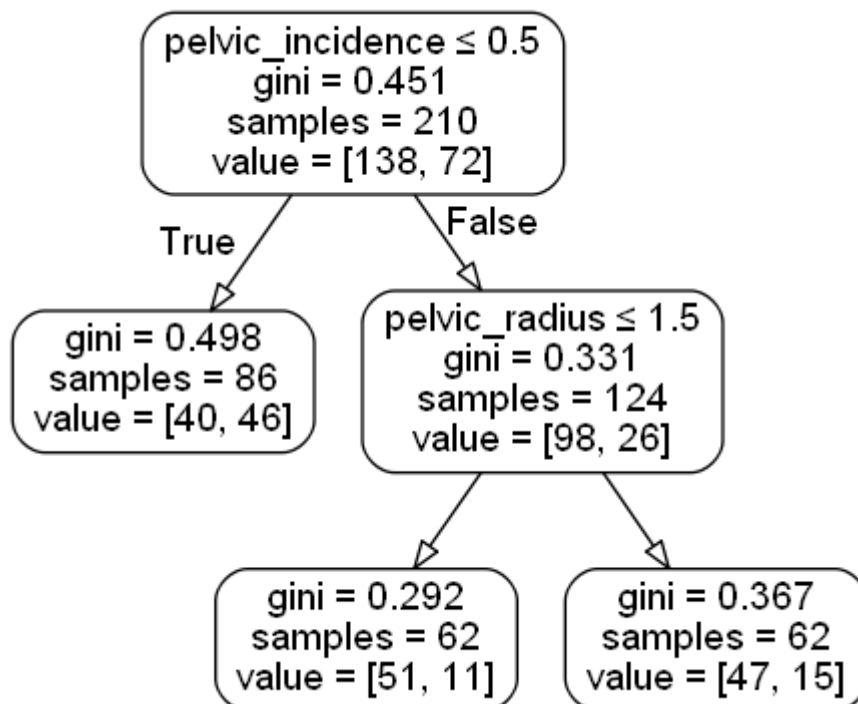
Decision Tree classifier for Minimum 12 records at leaf nodes



Decision Tree classifier for Minimum 30 records at leaf nodes



Decision Tree classifier for Minimum 50 records at leaf nodes



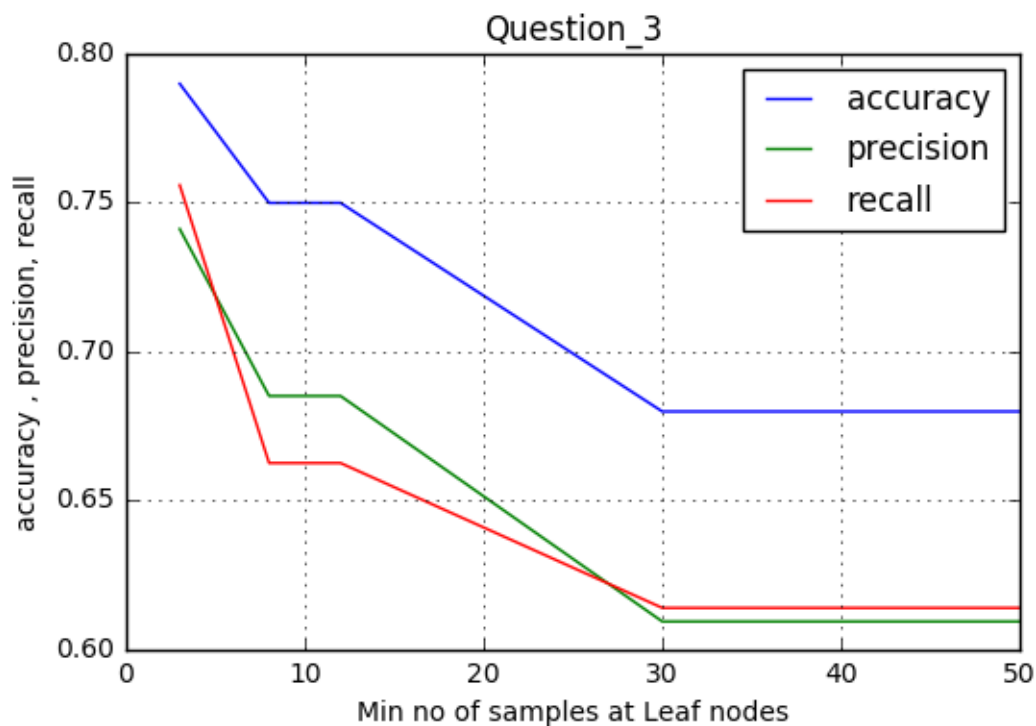
Accuracy, Precision, Recall for Dt with 3 data sets at leaf node are 0.79, 0.741234221598878, 0.7559523809523809

Accuracy, Precision, Recall for Dt with 8 data sets at leaf node are 0.75, 0.6852060982495765, 0.6626984126984128

Accuracy, Precision, Recall for Dt with 12 data sets at leaf node are 0.75, 0.6852060982495765, 0.6626984126984128

Accuracy, Precision, Recall for Dt with 30 data sets at leaf node are 0.68, 0.6095238095238096, 0.6140873015873016

Accuracy, Precision, Recall for Dt with 50 data sets at leaf node are 0.68, 0.6095238095238096, 0.6140873015873016



Answer 3c:

When compared to the first question these metrics have lower overall performance and have the best performance at 3 minimum samples at leaf nodes. This change is because we have done binning and hence there are only few values, so the best split data will be different. We notice one more thing that is for both decision trees with 8 and 12 minimum samples the metrics are same, same in case with 30 and 50. The other most important thing noticed here is the more less the minimum number of samples, the better are the metrics for this case.

Important functions used and their syntaxes:

- 1) **train_test_split**(*arrays, **options) : Split arrays or matrices into random train and test subsets
- 2) **DecisionTreeClassifier**(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)
- 3) **export_graphviz**(decision_tree, out_file="tree.dot", max_depth=None, feature_names=None, class_names=None, label='all', filled=False, leaves_parallel=False, impurity=True, node_ids=False, proportion=False, rotate=False, rounded=False, special_characters=False, precision=3)

CODE: (USED IPYTHON – JUPYTER NOTEBOOK)

```
1. import pandas as pd
2. import numpy as np
3. from sklearn.cross_validation import train_test_split
4. from sklearn.tree import DecisionTreeClassifier
5. from sklearn.metrics import accuracy_score
6. from sklearn.metrics import precision_score
7. from sklearn.metrics import recall_score
8. from sklearn import tree
9. import matplotlib.pyplot as plt
10. import xlswriter
11. import pydotplus
12. from sklearn.metrics import confusion_matrix
13. from sklearn.metrics import precision_recall_fscore_support
14.
15.
16. def tree_generation_1(f_name):
17.     accuracy=[]
18.     precision=[]
19.     recall=[]
20.     min_datasets=[3,8,12,30,50]
21.
22.
23.     # reading the data using read_csv function of the pandas library which directly
24.     # reads the csv file and creates a datagram of the same
25.     data = pd.read_csv(f_name)
26.
27.     # dividing the data into two sets , ie X denotes all the attributes data and Y
28.     # denotes the class output data which is to be predicted
29.     X = data.values[:, 0:6]      #1st to 6th column
30.     Y = data.values[:,6]        #last column
31.
32.     # dividing the records into testing and training data using train_test_split fu
33.     # nction, which takes percentage as input for test size
34.     # hence 310 given samples , 210 to be used as test_size so it is 67.77%
35.     X_train, X_test, y_train, y_test = train_test_split( X, Y, train_size = 0.678,
36.     random_state =100)
37.
38.     #creating the classifier using DecisionTreeClassifier fucntion with Gini index
39.     # approach and minimum records per leaf node as 3
40.     dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=3)
41.     dt_gini.fit(X_train, y_train)
42.
43.     #predicting the class values by testing the model with testing dataset and stor
44.     # ing them
```

```

44.     y_pred_gini = dt_gini.predict(X_test)
45.
46.     #calculating the accuracy, precision and recal and storing them
47.     a = accuracy_score(y_test,y_pred_gini)
48.     b = precision_score(y_test, y_pred_gini,average='macro')
49.     c = recall_score(y_test, y_pred_gini,average='macro')
50.     print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are
    {},{},{}".format(3,a,b,c))
51.     accuracy.append(a)
52.     precision.append(b)
53.     recall.append(c)
54.
55.
56.     #exporting the created decision tree to pdf
57.     dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],
    out_file=None,filled=True,rounded=True,special_characters=True)
58.     graph = pydotplus.graph_from_dot_data(dot_data)
59.     nodes = graph.get_node_list()
60.     colors = ('blue', 'yellow', 'green', 'red', 'white')
61.     for node in nodes:
62.         if node.get_name() not in ('node', 'edge'):
63.             values = dt_gini.tree_.value[int(node.get_name())][0]
64.             #color only nodes where only one class is present
65.             if max(values) == sum(values):
66.                 node.set_fillcolor(colors[np.argmax(values)])
67.             #mixed nodes get the default color
68.             else:
69.                 node.set_fillcolor(colors[-1])
70.
71.     graph.write_png('dt_2c_3min.png')
72.
73.     #creating the classifier using DecisionTreeClassifier fucntion with Gini index
    approach and minimum records per leaf node as 8
74.     dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=8)
75.     dt_gini.fit(X_train, y_train)
76.
77.
78.     #predicting the class values by testing the model with testing dataset and stor
    ing them
79.     y_pred_gini = dt_gini.predict(X_test)
80.
81.
82.     #calculating the accuracy, precision and recal and storing them
83.     a = accuracy_score(y_test,y_pred_gini)
84.     b = precision_score(y_test, y_pred_gini,average='macro')
85.     c = recall_score(y_test, y_pred_gini,average='macro')
86.     print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are
    {},{},{}".format(8,a,b,c))
87.     accuracy.append(a)
88.     precision.append(b)
89.     recall.append(c)
90.
91.
92.     #exporting the created decision tree to pdf
93.     dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],
    out_file=None,filled=True,rounded=True,special_characters=True)
94.     graph = pydotplus.graph_from_dot_data(dot_data)
95.     nodes = graph.get_node_list()
96.     colors = ('blue', 'yellow', 'green', 'red', 'white')
97.     for node in nodes:
98.         if node.get_name() not in ('node', 'edge'):
99.             values = dt_gini.tree_.value[int(node.get_name())][0]
100.            #color only nodes where only one class is present
101.            if max(values) == sum(values):
102.                node.set_fillcolor(colors[np.argmax(values)])
103.            #mixed nodes get the default color

```

```

104.         else:
105.             node.set_fillcolor(colors[-1])
106.
107.         graph.write_png('dt_2c_8min.png')
108.
109.
110.
111.         #creating the classifier using DecisionTreeClassifier fucntion with Gini
index approach and minimum records per leaf node as 12
112.         dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=12)
113.
114.         dt_gini.fit(X_train, y_train)
115.
116.
117.         #predicting the class values by testing the model with testing dataset a
nd storing them
118.         y_pred_gini = dt_gini.predict(X_test)
119.
120.
121.         #calculating the accuracy, precision and recal and storing them
122.         a = accuracy_score(y_test,y_pred_gini)
123.         b = precision_score(y_test, y_pred_gini,average='macro')
124.         c = recall_score(y_test, y_pred_gini,average='macro')
125.         print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf n
ode are {},{},{},{}".format(12,a,b,c))
126.         accuracy.append(a)
127.         precision.append(b)
128.         recall.append(c)
129.
130.
131.         #exporting the created decision tree to pdf
132.         dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.value
s[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
133.         graph = pydotplus.graph_from_dot_data(dot_data)
134.         nodes = graph.get_node_list()
135.         colors = ('blue', 'yellow', 'green', 'red', 'white')
136.         for node in nodes:
137.             if node.get_name() not in ('node', 'edge'):
138.                 values = dt_gini.tree_.value[int(node.get_name())][0]
139.                 #color only nodes where only one class is present
140.                 if max(values) == sum(values):
141.                     node.set_fillcolor(colors[np.argmax(values)])
142.                 #mixed nodes get the default color
143.                 else:
144.                     node.set_fillcolor(colors[-1])
145.
146.         graph.write_png('dt_2c_12min.png')
147.
148.
149.         #creating the classifier using DecisionTreeClassifier fucntion with Gini
index approach and minimum records per leaf node as 30
150.         dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=30)
151.
152.         dt_gini.fit(X_train, y_train)
153.
154.         #predicting the class values by testing the model with testing dataset a
nd storing them
155.         y_pred_gini = dt_gini.predict(X_test)
156.
157.         #calculating the accuracy, precision and recal and storing them
158.         a = accuracy_score(y_test,y_pred_gini)
159.         b = precision_score(y_test, y_pred_gini,average='macro')
160.         c = recall_score(y_test, y_pred_gini,average='macro')

```



```

161.         print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf n
ode are {},{},{}".format(30,a,b,c))
162.         accuracy.append(a)
163.         precision.append(b)
164.         recall.append(c)
165.
166.         #exporting the created decision tree to pdf
167.         with open("biomechanical_dt_gini_30.dot", "w") as f:
168.             f = tree.export_graphviz(dt_gini, out_file=f,feature_names=data.columns
.values[0:6])
169.             !dot -Tpdf "biomechanical_dt_gini_30.dot" -
o "biomechanical_dt_gini_30.pdf #exporting the created decision tree to pdf
170.             dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.value
s[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
171.             graph = pydotplus.graph_from_dot_data(dot_data)
172.             nodes = graph.get_node_list()
173.             colors = ('blue', 'yellow', 'green', 'red', 'white')
174.             for node in nodes:
175.                 if node.get_name() not in ('node', 'edge'):
176.                     values = dt_gini.tree_.value[int(node.get_name())][0]
177.                     #color only nodes where only one class is present
178.                     if max(values) == sum(values):
179.                         node.set_fillcolor(colors[np.argmax(values)])
180.                     #mixed nodes get the default color
181.                     else:
182.                         node.set_fillcolor(colors[-1])
183.
184.             graph.write_png('dt_2c_30min.png')
185.
186.
187.         #creating the classifier using DecisionTreeClassifier fuction with Gini
index approach and minimum records per leaf node as 50
188.         dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=50)
189.
190.         dt_gini.fit(X_train, y_train)
191.
192.
193.         #predicting the class values by testing the model with testing dataset a
nd storing them
194.         y_pred_gini = dt_gini.predict(X_test)
195.
196.         #calculating the accuracy, precision and recal and storing them
197.         a = accuracy_score(y_test,y_pred_gini)
198.         b = precision_score(y_test, y_pred_gini,average='macro')
199.         c = recall_score(y_test, y_pred_gini,average='macro')
200.         print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf n
ode are {},{},{}".format(50,a,b,c))
201.         accuracy.append(a)
202.         precision.append(b)
203.         recall.append(c)
204.
205.         #exporting the created decision tree to pdf
206.         dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.value
s[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
207.         graph = pydotplus.graph_from_dot_data(dot_data)
208.         nodes = graph.get_node_list()
209.         colors = ('blue', 'yellow', 'green', 'red', 'white')
210.         for node in nodes:
211.             if node.get_name() not in ('node', 'edge'):
212.                 values = dt_gini.tree_.value[int(node.get_name())][0]
213.                 #color only nodes where only one class is present
214.                 if max(values) == sum(values):
215.                     node.set_fillcolor(colors[np.argmax(values)])
216.                 #mixed nodes get the default color
217.                 else:

```

```

218.         node.set_fillcolor(colors[-1])
219.
220.         graph.write_png('dt_2c_50min.png')
221.
222.     #plotting graphs
223.     fig1 = plt.figure(1)
224.     plt.plot(min_datasets,accuracy,label="accuracy")
225.     plt.plot(min_datasets,precision,label="precision")
226.     plt.plot(min_datasets,recall,label="recall")
227.     plt.grid(True)
228.     plt.xlabel('Min no of samples at Leaf nodes')
229.     plt.ylabel('accuracy , precision, recall')
230.     plt.legend(['accuracy', 'precision', 'recall'], loc='upper right')
231.     plt.title('Question_1')
232.     fig1.savefig('plot_1.png')
233.     plt.close()
234.     # add plt.close() after you've saved the figure
235.     #plt.show()
236.
237.
238.     fig2 = plt.figure(2)
239.     plt.plot(precision,recall,'ro')
240.     plt.xlabel('precision')
241.     plt.ylabel('recall')
242.     plt.title('Question_1 precision vs recall')
243.     fig2.savefig('precision_recall_1.png')
244.     plt.close()
245.     print('\n ##### \n')
246.
247.     def tree_generation_2(f_name):
248.         accuracy=[]
249.         precision=[]
250.         recall=[]
251.         precision_class_normal=[]
252.         recall_class_normal=[]
253.         precision_class_spondy=[]
254.         recall_class_spondy=[]
255.         precision_class_hernia=[]
256.         recall_class_hernia=[]
257.         min_datasets=[3,8,12,30,50]
258.
259.
260.         # reading the data using read_csv function of the pandas library which d
261.         irectly reads the csv file and creates a datagram of the same
262.         data = pd.read_csv(f_name)
263.
264.         # dividing the data into two sets , ie X denotes all the attributes data
265.         and Y denotes the class output data which is to be predicted
266.         X = data.values[:, 0:6]      #1st to 6th column
267.         Y = data.values[:,6]         #last column
268.
269.         # dividing the records into testing and training data using train_test_s
270.         plit function, which takes percentage as input for test size
271.         # hence 310 given samples , 210 to be used as test_size so it is 67.77%
272.
273.         X_train, X_test, y_train, y_test = train_test_split( X, Y, train_size =
274.         0.678, random_state = 60)
275.
276.         #creating the classifier using DecisionTreeClassifier fucntion with Gini
277.         index approach and minimum records per leaf node as 3
278.         dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=3)

```

```

276.         dt_gini.fit(X_train, y_train)
277.
278.
279.         #predicting the class values by testing the model with testing dataset a
nd storing them
280.         y_pred_gini = dt_gini.predict(X_test)
281.
282.         #calculating the accuracy, precision and recal and storing them
283.         a = accuracy_score(y_test,y_pred_gini)
284.         precision_recall=precision_recall_fscore_support(y_test, y_pred_gini, av
erage=None,labels=['Normal', 'Spondylolisthesis', 'Hernia'])
285.         b=precision_recall[0][0]
286.         c=precision_recall[0][1]
287.         d=precision_recall[1][0]
288.         e=precision_recall[1][1]
289.         f=precision_recall[2][0]
290.         g=precision_recall[2][1]
291.         print("Accuracy for gini Dt with {} data sets at leaf node are {}".forma
t(3,a))
292.         print("precision , recall for class normal is {}, {}".format(b,c))
293.         print("precision , recall for class spondy is {}, {}".format(d,e))
294.         print("precision , recall for class hernia is {}, {} \n".format(f,g))
295.         accuracy.append(a)
296.         precision_class_normal.append(b)
297.         recall_class_normal.append(c)
298.         precision_class_spondy.append(d)
299.         recall_class_spondy.append(e)
300.         precision_class_hernia.append(f)
301.         recall_class_hernia.append(g)
302.
303.
304.
305.         #exporting the created decision tree to pdf
306.         dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.value
s[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
307.         graph = pydotplus.graph_from_dot_data(dot_data)
308.         nodes = graph.get_node_list()
309.         colors = ('blue', 'yellow', 'green', 'red', 'white')
310.         for node in nodes:
311.             if node.get_name() not in ('node', 'edge'):
312.                 values = dt_gini.tree_.value[int(node.get_name())][0]
313.                 #color only nodes where only one class is present
314.                 if max(values) == sum(values):
315.                     node.set_fillcolor(colors[np.argmax(values)])
316.                 #mixed nodes get the default color
317.                 else:
318.                     node.set_fillcolor(colors[-1])
319.
320.         graph.write_png('dt_3c_3min.png')
321.
322.
323.         #creating the classifier using DecisionTreeClassifier fucntion with Gini
index approach and minimum records per leaf node as 8
324.         dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=8)
325.
326.         dt_gini.fit(X_train, y_train)
327.
328.         #predicting the class values by testing the model with testing dataset a
nd storing them
329.         y_pred_gini = dt_gini.predict(X_test)
330.
331.         #calculating the accuracy, precision and recal and storing them
332.         a = accuracy_score(y_test,y_pred_gini)
333.         precision_recall=precision_recall_fscore_support(y_test, y_pred_gini, av
erage=None,labels=['Normal', 'Spondylolisthesis', 'Hernia'])

```

```

334.         b=precision_recall[0][0]
335.         c=precision_recall[0][1]
336.         d=precision_recall[1][0]
337.         e=precision_recall[1][1]
338.         f=precision_recall[2][0]
339.         g=precision_recall[2][1]
340.         print("Accuracy for gini Dt with {} data sets at leaf node are {}".forma
t(8,a))
341.         print("precision , recall for class normal is {}, {}".format(b,c))
342.         print("precision , recall for class spondy is {}, {}".format(d,e))
343.         print("precision , recall for class hernia is {}, {} \n".format(f,g))
344.         accuracy.append(a)
345.         precision_class_normal.append(b)
346.         recall_class_normal.append(c)
347.         precision_class_spondy.append(d)
348.         recall_class_spondy.append(e)
349.         precision_class_hernia.append(f)
350.         recall_class_hernia.append(g)
351.
352.
353.         #exporting the created decision tree to pdf
354.         dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.value
s[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
355.         graph = pydotplus.graph_from_dot_data(dot_data)
356.         nodes = graph.get_node_list()
357.         colors = ('blue', 'yellow', 'green', 'red', 'white')
358.         for node in nodes:
359.             if node.get_name() not in ('node', 'edge'):
360.                 values = dt_gini.tree_.value[int(node.get_name())][0]
361.                 #color only nodes where only one class is present
362.                 if max(values) == sum(values):
363.                     node.set_fillcolor(colors[np.argmax(values)])
364.                 #mixed nodes get the default color
365.                 else:
366.                     node.set_fillcolor(colors[-1])
367.
368.         graph.write_png('dt_3c_8min.png')
369.
370.
371.         #creating the classifier using DecisionTreeClassifier fucntion with Gini
index approach and minimum records per leaf node as 12
372.         dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=12)
373.
374.         dt_gini.fit(X_train, y_train)
375.
376.         #predicting the class values by testing the model with testing dataset a
nd storing them
377.         y_pred_gini = dt_gini.predict(X_test)
378.
379.         #calculating the accuracy, precision and recal and storing them
380.         a = accuracy_score(y_test,y_pred_gini)
381.         precision_recall=precision_recall_fscore_support(y_test, y_pred_gini, av
erage=None,labels=['Normal','Spondylolisthesis','Hernia'])
382.         b=precision_recall[0][0]
383.         c=precision_recall[0][1]
384.         d=precision_recall[1][0]
385.         e=precision_recall[1][1]
386.         f=precision_recall[2][0]
387.         g=precision_recall[2][1]
388.         print("Accuracy for gini Dt with {} data sets at leaf node are {}".forma
t(12,a))
389.         print("precision , recall for class normal is {}, {}".format(b,c))
390.         print("precision , recall for class spondy is {}, {}".format(d,e))
391.         print("precision , recall for class hernia is {}, {} \n".format(f,g))
392.         accuracy.append(a)

```

```

393.         precision_class_normal.append(b)
394.         recall_class_normal.append(c)
395.         precision_class_spondy.append(d)
396.         recall_class_spondy.append(e)
397.         precision_class_hernia.append(f)
398.         recall_class_hernia.append(g)
399.
400.
401.         #exporting the created decision tree to pdf
402.         dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,
filled=True,rounded=True,special_characters=True)
403.         graph = pydotplus.graph_from_dot_data(dot_data)
404.         nodes = graph.get_node_list()
405.         colors = ('blue', 'yellow', 'green', 'red', 'white')
406.         for node in nodes:
407.             if node.get_name() not in ('node', 'edge'):
408.                 values = dt_gini.tree_.value[int(node.get_name())][0]
409.                 #color only nodes where only one class is present
410.                 if max(values) == sum(values):
411.                     node.set_fillcolor(colors[np.argmax(values)])
412.                     #mixed nodes get the default color
413.                 else:
414.                     node.set_fillcolor(colors[-1])
415.
416.         graph.write_png('dt_3c_12min.png')
417.
418.
419.         #creating the classifier using DecisionTreeClassifier function with Gini
index approach and minimum records per leaf node as 30
420.         dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=30)
421.
422.         dt_gini.fit(X_train, y_train)
423.
424.         #predicting the class values by testing the model with testing dataset and
storing them
425.         y_pred_gini = dt_gini.predict(X_test)
426.
427.         #calculating the accuracy, precision and recall and storing them
428.         a = accuracy_score(y_test,y_pred_gini)
429.         precision_recall=precision_recall_fscore_support(y_test, y_pred_gini, average=None,labels=['Normal','Spondylolisthesis','Hernia'])
430.         b=precision_recall[0][0]
431.         c=precision_recall[0][1]
432.         d=precision_recall[1][0]
433.         e=precision_recall[1][1]
434.         f=precision_recall[2][0]
435.         g=precision_recall[2][1]
436.         print("Accuracy for gini Dt with {} data sets at leaf node are {}".format(30,a))
437.         print("precision , recall for class normal is {}, {}".format(b,c))
438.         print("precision , recall for class spondy is {}, {}".format(d,e))
439.         print("precision , recall for class hernia is {}, {} \n".format(f,g))
440.         accuracy.append(a)
441.         precision_class_normal.append(b)
442.         recall_class_normal.append(c)
443.         precision_class_spondy.append(d)
444.         recall_class_spondy.append(e)
445.         precision_class_hernia.append(f)
446.         recall_class_hernia.append(g)
447.
448.         #exporting the created decision tree to pdf
449.         dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,
filled=True,rounded=True,special_characters=True)
450.         graph = pydotplus.graph_from_dot_data(dot_data)
451.         nodes = graph.get_node_list()

```

```

452.         colors = ('blue', 'yellow', 'green', 'red', 'white')
453.         for node in nodes:
454.             if node.get_name() not in ('node', 'edge'):
455.                 values = dt_gini.tree_.value[int(node.get_name())][0]
456.                 #color only nodes where only one class is present
457.                 if max(values) == sum(values):
458.                     node.set_fillcolor(colors[np.argmax(values)])
459.                 #mixed nodes get the default color
460.                 else:
461.                     node.set_fillcolor(colors[-1])
462.
463.         graph.write_png('dt_3c_30min.png')
464.
465.
466.
467.         #creating the classifier using DecisionTreeClassifier function with Gini
index approach and minimum records per leaf node as 50
468.         dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=50)
469.
470.         dt_gini.fit(X_train, y_train)
471.
472.
473.         #predicting the class values by testing the model with testing dataset a
nd storing them
474.         y_pred_gini = dt_gini.predict(X_test)
475.
476.         #calculating the accuracy, precision and recal and storing them
477.         a = accuracy_score(y_test,y_pred_gini)
478.         precision_recall=precision_recall_fscore_support(y_test, y_pred_gini, av
erage=None,labels=['Normal','Spondylolisthesis','Hernia'])
479.         b=precision_recall[0][0]
480.         c=precision_recall[0][1]
481.         d=precision_recall[1][0]
482.         e=precision_recall[1][1]
483.         f=precision_recall[2][0]
484.         g=precision_recall[2][1]
485.         print("Accuracy for gini Dt with {} data sets at leaf node are {}".forma
t(50,a))
486.         print("precision , recall for class normal is {}, {}".format(b,c))
487.         print("precision , recall for class Spondy is {}, {}".format(d,e))
488.         print("precision , recall for class hernia is {}, {} \n".format(f,g))
489.         accuracy.append(a)
490.         precision_class_normal.append(b)
491.         recall_class_normal.append(c)
492.         precision_class_spondy.append(d)
493.         recall_class_spondy.append(e)
494.         precision_class_hernia.append(f)
495.         recall_class_hernia.append(g)
496.
497.
498.         #exporting the created decision tree to pdf
499.         dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.value
s[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
500.         graph = pydotplus.graph_from_dot_data(dot_data)
501.         nodes = graph.get_node_list()
502.         colors = ('blue', 'yellow', 'green', 'red', 'white')
503.         for node in nodes:
504.             if node.get_name() not in ('node', 'edge'):
505.                 values = dt_gini.tree_.value[int(node.get_name())][0]
506.                 #color only nodes where only one class is present
507.                 if max(values) == sum(values):
508.                     node.set_fillcolor(colors[np.argmax(values)])
509.                 #mixed nodes get the default color
510.                 else:
511.                     node.set_fillcolor(colors[-1])

```

```

512.
513.         graph.write_png('dt_3c_50min.png')
514.
515.         #plotting graphs
516.         fig1 = plt.figure(1)
517.         ax = plt.subplot(111)
518.         ax.plot(min_datasets,accuracy,label="accuracy")
519.         ax.plot(min_datasets,precision_class_normal,label="precision_normal")
520.         ax.plot(min_datasets,recall_class_normal,label="recall_normal")
521.         ax.plot(min_datasets,precision_class_spondy,label="precision_spondy")
522.         ax.plot(min_datasets,recall_class_spondy,label="recall_spondy")
523.         ax.plot(min_datasets,precision_class_hernia,label="precision_hernia")
524.         ax.plot(min_datasets,recall_class_hernia,label="recall_hernia")
525.         ax.grid(True)
526.         ax.set_xlabel('Min no of samples at Leaf nodes')
527.         #plt.ylabel
528.         box = ax.get_position()
529.         ax.set_position([box.x0, box.y0 + box.height * 0.2,box.width, box.height
530. * 0.9])
531.
532.         ax.legend(['accuracy', 'precision_normal', 'recall_normal','precision_spon
533. dy','recall_spondy','precision_hernia','recall_hernia'],loc='upper center', bbox_
534. to_anchor=(0.5, -0.05),
535.                 ncol=3)
536.         fig1.savefig('plot_2.png')
537.         plt.close()
538.         # add plt.close() after you've saved the figure
539.         #plt.show()
540.
541.         print('\n #####
542. ##### \n')
543.
544.         def tree_generation_3(f_name):
545.             df = pd.read_csv(f_name)
546.             revised_dataframe = pd.DataFrame()
547.             bin_boundary=[]
548.             columns=df.columns
549.             y=0
550.             for col in columns[0:6]:
551.                 hist, bin_edges = np.histogram(df[col][1:], bins=4)
552.                 bin_boundary.append(bin_edges)
553.                 col_trans_output=pd.DataFrame(pd.cut(df[col],4,labels=range(4)))
554.                 revised_dataframe = pd.concat([revised_dataframe,col_trans_output],
555. axis=1)
556.
557.             writer = pd.ExcelWriter('test1.xlsx',engine='xlsxwriter')
558.             workbook=writer.book
559.             revised_dataframe.to_excel(writer,sheet_name='Validation1',startrow=0 ,
560. startcol=0)
561.             workbook.close()
562.
563.             for x in bin_boundary:
564.                 print('Boundaries for {} are {} \n'.format(columns[y],x))
565.                 y=y+1
566.             print('\n #####
567. ##### \n')
568.
569.             accuracy=[]
570.             precision=[]
571.             recall=[]
572.             min_datasets=[3,8,12,30,50]
573.
574.             data=pd.read_csv(f_name)
575.             #print(revised_dataframe)

```

```

571.         # dividing the data into two sets , ie X denotes all the attributes data
           and Y denotes the class output data which is to be predicted
572.         X = revised_dataframe.values[:, 0:6]          #1st to 6th column
573.         Y = df.values[:,6]                          #last column
574.
575.
576.         # dividing the records into testing and training data using train_test_s
           plit function, which takes percentage as input for test size
577.         # hence 310 given samples , 210 to be used as test_size so it is 67.77%

578.         X_train, X_test, y_train, y_test = train_test_split( X, Y, train_size =
           0.678, random_state = 42)
579.
580.
581.
582.         #creating the classifier using DecisionTreeClassifier fucntion with Gini
           index approach and minimum records per leaf node as 3
583.         dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=3)

584.         dt_gini.fit(X_train, y_train)
585.
586.
587.         #predicting the class values by testing the model with testing dataset a
           nd storing them
588.         y_pred_gini = dt_gini.predict(X_test)
589.
590.         #calculating the accuracy, precision and recal and storing them
591.         a = accuracy_score(y_test,y_pred_gini)
592.         b = precision_score(y_test, y_pred_gini,average='macro')
593.         c = recall_score(y_test, y_pred_gini,average='macro')
594.         print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf n
           ode are {},{},{},{}".format(3,a,b,c))
595.         accuracy.append(a)
596.         precision.append(b)
597.         recall.append(c)
598.
599.
600.         #exporting the created decision tree to pdf
601.         dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.value
           s[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
602.         graph = pydotplus.graph_from_dot_data(dot_data)
603.         nodes = graph.get_node_list()
604.         colors = ('blue', 'yellow', 'green', 'red', 'white')
605.         for node in nodes:
606.             if node.get_name() not in ('node', 'edge'):
607.                 values = dt_gini.tree_.value[int(node.get_name())][0]
608.                 #color only nodes where only one class is present
609.                 if max(values) == sum(values):
610.                     node.set_fillcolor(colors[np.argmax(values)])
611.                 #mixed nodes get the default color
612.                 else:
613.                     node.set_fillcolor(colors[-1])
614.
615.         graph.write_png('dt_rev_3min.png')
616.
617.
618.         #creating the classifier using DecisionTreeClassifier fucntion with Gini
           index approach and minimum records per leaf node as 8
619.         dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=8)

620.         dt_gini.fit(X_train, y_train)
621.
622.
623.         #predicting the class values by testing the model with testing dataset a
           nd storing them
624.         y_pred_gini = dt_gini.predict(X_test)

```



```

625.
626.
627.     #calculating the accuracy, precision and recal and storing them
628.     a = accuracy_score(y_test,y_pred_gini)
629.     b = precision_score(y_test, y_pred_gini,average='macro')
630.     c = recall_score(y_test, y_pred_gini,average='macro')
631.     print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf n
ode are {},{},{}".format(8,a,b,c))
632.     accuracy.append(a)
633.     precision.append(b)
634.     recall.append(c)
635.
636.
637.     #exporting the created decision tree to pdf
638.     dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.value
s[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
639.     graph = pydotplus.graph_from_dot_data(dot_data)
640.     nodes = graph.get_node_list()
641.     colors = ('blue', 'yellow', 'green', 'red', 'white')
642.     for node in nodes:
643.         if node.get_name() not in ('node', 'edge'):
644.             values = dt_gini.tree_.value[int(node.get_name())][0]
645.             #color only nodes where only one class is present
646.             if max(values) == sum(values):
647.                 node.set_fillcolor(colors[np.argmax(values)])
648.             #mixed nodes get the default color
649.             else:
650.                 node.set_fillcolor(colors[-1])
651.
652.     graph.write_png('dt_rev_8min.png')
653.
654.
655.
656.     #creating the classifier using DecisionTreeClassifier fucntion with Gini
index approach and minimum records per leaf node as 12
657.     dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=12)
658.
659.     dt_gini.fit(X_train, y_train)
660.
661.
662.     #predicting the class values by testing the model with testing dataset a
nd storing them
663.     y_pred_gini = dt_gini.predict(X_test)
664.
665.
666.     #calculating the accuracy, precision and recal and storing them
667.     a = accuracy_score(y_test,y_pred_gini)
668.     b = precision_score(y_test, y_pred_gini,average='macro')
669.     c = recall_score(y_test, y_pred_gini,average='macro')
670.     print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf n
ode are {},{},{}".format(12,a,b,c))
671.     accuracy.append(a)
672.     precision.append(b)
673.     recall.append(c)
674.
675.
676.     #exporting the created decision tree to pdf
677.     dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.value
s[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
678.     graph = pydotplus.graph_from_dot_data(dot_data)
679.     nodes = graph.get_node_list()
680.     colors = ('blue', 'yellow', 'green', 'red', 'white')
681.     for node in nodes:
682.         if node.get_name() not in ('node', 'edge'):
683.             values = dt_gini.tree_.value[int(node.get_name())][0]

```

```

684.         #color only nodes where only one class is present
685.         if max(values) == sum(values):
686.             node.set_fillcolor(colors[np.argmax(values)])
687.         #mixed nodes get the default color
688.         else:
689.             node.set_fillcolor(colors[-1])
690.
691.     graph.write_png('dt_rev_12min.png')
692.
693.
694.     #creating the classifier using DecisionTreeClassifier fucntion with Gini
index approach and minimum records per leaf node as 30
695.     dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=30)
696.
697.     dt_gini.fit(X_train, y_train)
698.
699.     #predicting the class values by testing the model with testing dataset a
nd storing them
700.     y_pred_gini = dt_gini.predict(X_test)
701.
702.     #calculating the accuracy, precision and recal and storing them
703.     a = accuracy_score(y_test,y_pred_gini)
704.     b = precision_score(y_test, y_pred_gini,average='macro')
705.     c = recall_score(y_test, y_pred_gini,average='macro')
706.     print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf n
ode are {},{},{},{}".format(30,a,b,c))
707.     accuracy.append(a)
708.     precision.append(b)
709.     recall.append(c)
710.
711.     #exporting the created decision tree to pdf
712.     dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.value
s[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
713.     graph = pydotplus.graph_from_dot_data(dot_data)
714.     nodes = graph.get_node_list()
715.     colors = ('blue', 'yellow', 'green', 'red', 'white')
716.     for node in nodes:
717.         if node.get_name() not in ('node', 'edge'):
718.             values = dt_gini.tree_.value[int(node.get_name())][0]
719.             #color only nodes where only one class is present
720.             if max(values) == sum(values):
721.                 node.set_fillcolor(colors[np.argmax(values)])
722.             #mixed nodes get the default color
723.             else:
724.                 node.set_fillcolor(colors[-1])
725.
726.     graph.write_png('dt_rev_30min.png')
727.
728.
729.
730.     #creating the classifier using DecisionTreeClassifier fucntion with Gini
index approach and minimum records per leaf node as 50
731.     dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=50)
732.
733.     dt_gini.fit(X_train, y_train)
734.
735.
736.     #predicting the class values by testing the model with testing dataset a
nd storing them
737.     y_pred_gini = dt_gini.predict(X_test)
738.
739.     #calculating the accuracy, precision and recal and storing them
740.     a = accuracy_score(y_test,y_pred_gini)
741.     b = precision_score(y_test, y_pred_gini,average='macro')

```

```

742.         c = recall_score(y_test, y_pred_gini,average='macro')
743.         print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf n
ode are {},{},{},{}".format(50,a,b,c))
744.         accuracy.append(a)
745.         precision.append(b)
746.         recall.append(c)
747.
748.
749.         #exporting the created decision tree to pdf
750.         dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.value
s[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
751.         graph = pydotplus.graph_from_dot_data(dot_data)
752.         nodes = graph.get_node_list()
753.         colors = ('blue', 'yellow', 'green', 'red', 'white')
754.         for node in nodes:
755.             if node.get_name() not in ('node', 'edge'):
756.                 values = dt_gini.tree_.value[int(node.get_name())][0]
757.                 #color only nodes where only one class is present
758.                 if max(values) == sum(values):
759.                     node.set_fillcolor(colors[np.argmax(values)])
760.                 #mixed nodes get the default color
761.                 else:
762.                     node.set_fillcolor(colors[-1])
763.
764.         graph.write_png('dt_rev_50min.png')
765.
766.         #plotting graphs
767.         fig1 = plt.figure(1)
768.         plt.plot(min_datasets,accuracy,label="accuracy")
769.         plt.plot(min_datasets,precision,label="precision")
770.         plt.plot(min_datasets,recall,label="recall")
771.         plt.grid(True)
772.         plt.xlabel('Min no of samples at Leaf nodes')
773.         plt.ylabel('accuracy , precision, recall')
774.         plt.legend(['accuracy', 'precision', 'recall'], loc='upper right')
775.         plt.title('Question_3')
776.         fig1.savefig('plot_3.png')
777.         plt.close()
778.         # add plt.close() after you've saved the figure
779.         #plt.show()
780.
781.
782.         fig2 = plt.figure(2)
783.         plt.plot(precision,recall,'ro')
784.         plt.xlabel('precision')
785.         plt.ylabel('recall')
786.         plt.title('Question_3 precision vs recall')
787.         fig2.savefig('precision_recall_3.png')
788.         plt.close()
789.         tree_generation_1("Biomechanical_Data_column_2C_weka.csv")
790.         tree_generation_2("BiomechanicalData_column_3C_weka.csv")
791.         tree_generation_3("Biomechanical_Data_column_2C_weka.csv")

```

Output:

```

Accuracy,Precision,Recall for gini Dt with 3 data sets at leaf node are
0.77,0.7473958333333333,0.7505494505494505
Accuracy,Precision,Recall for gini Dt with 8 data sets at leaf node are
0.78,0.7606358111266948,0.7450549450549451
Accuracy,Precision,Recall for gini Dt with 12 data sets at leaf node ar
e 0.79,0.7738095238095237,0.7527472527472527
Accuracy,Precision,Recall for gini Dt with 30 data sets at leaf node ar
e 0.73,0.7083333333333333,0.7197802197802198

```

Accuracy,Precision,Recall for gini Dt with 50 data sets at leaf node are 0.7,0.6671341748480599,0.6571428571428571

#####

Accuracy for gini Dt with 3 data sets at leaf node are 0.83
precision , recall for class normal is 0.7407407407407407, 0.94
precision , recall for class spondy is 0.7142857142857143, 0.9215686274509803
precision , recall for class hernia is 0.7272727272727273, 0.9306930693069307

Accuracy for gini Dt with 8 data sets at leaf node are 0.86
precision , recall for class normal is 0.7333333333333333, 1.0
precision , recall for class spondy is 0.7857142857142857, 0.9803921568627451
precision , recall for class hernia is 0.7586206896551724, 0.99009900990099

Accuracy for gini Dt with 12 data sets at leaf node are 0.82
precision , recall for class normal is 0.7083333333333334, 1.0
precision , recall for class spondy is 0.6071428571428571, 0.9803921568627451
precision , recall for class hernia is 0.6538461538461539, 0.99009900990099

Accuracy for gini Dt with 30 data sets at leaf node are 0.77
precision , recall for class normal is 0.5882352941176471, 1.0
precision , recall for class spondy is 0.7142857142857143, 0.9803921568627451
precision , recall for class hernia is 0.6451612903225806, 0.99009900990099

Accuracy for gini Dt with 50 data sets at leaf node are 0.83
precision , recall for class normal is 0.7037037037037037, 1.0
precision , recall for class Spondy is 0.6785714285714286, 0.9803921568627451
precision , recall for class hernia is 0.6909090909090909, 0.99009900990099

#####

Boundaries for pelvic_incidence are [26.14792141 52.06945121 77.9
9098101 103.9125108 129.8340406]

Boundaries for pelvic_tilt numeric are [-6.55494835 7.44175464 21.4
3845763 35.43516061 49.4318636]

Boundaries for lumbar_lordosis_angle are [14. 41.93559638
69.87119275 97.80678912 125.7423855]

Boundaries for sacral_slope are [13.3669307 40.38258942 67.39824
815 94.41390687 121.4295656]

Boundaries for pelvic_radius are [70.08257486 93.32969127 116.5768
0768 139.82392409 163.0710405]

Boundaries for degree_spondylolisthesis are [-11.05817866 96.3421365
3 203.74245172 311.14276691 418.5430821]

#####

Accuracy,Precision,Recall for gini Dt with 3 data sets at leaf node are
0.79,0.741234221598878,0.7559523809523809

Accuracy,Precision,Recall for gini Dt with 8 data sets at leaf node are
0.75,0.6852060982495765,0.6626984126984128

Accuracy,Precision,Recall for gini Dt with 12 data sets at leaf node ar
e 0.75,0.6852060982495765,0.6626984126984128

Accuracy,Precision,Recall for gini Dt with 30 data sets at leaf node ar
e 0.68,0.6095238095238096,0.6140873015873016

Accuracy,Precision,Recall for gini Dt with 50 data sets at leaf node ar
e 0.68,0.6095238095238096,0.6140873015873016