INTELLIGENT DATA ANALYSIS – SPRING 2525

M13254448

Technical details:

Scripting language : Python

Packages used : Sklearn (scikit learn), matplotlib and pandas

Platform used: Jupyter notebook

QUESTION 1 :

Create decision trees using training set and set the 'minimum number of leaf nodes' to be 5,15,25,40 and 50.

Solution:

The decision trees generated are as follows, in the ascending order of minimum number of leaf nodes. This is using 'GINI' index as the criterion for splitting.

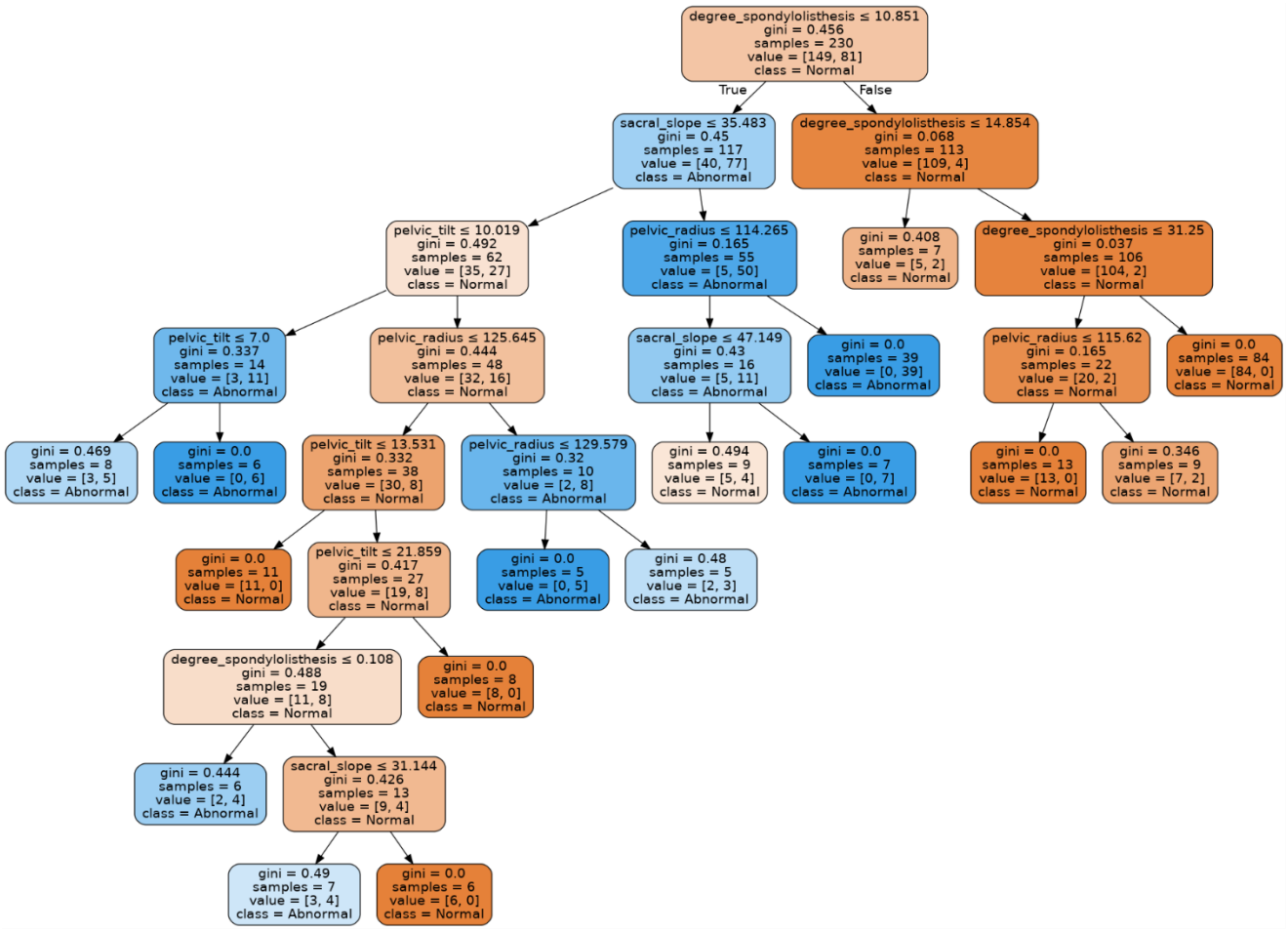


Fig 1(a) : Minimum number of leaf nodes = 5

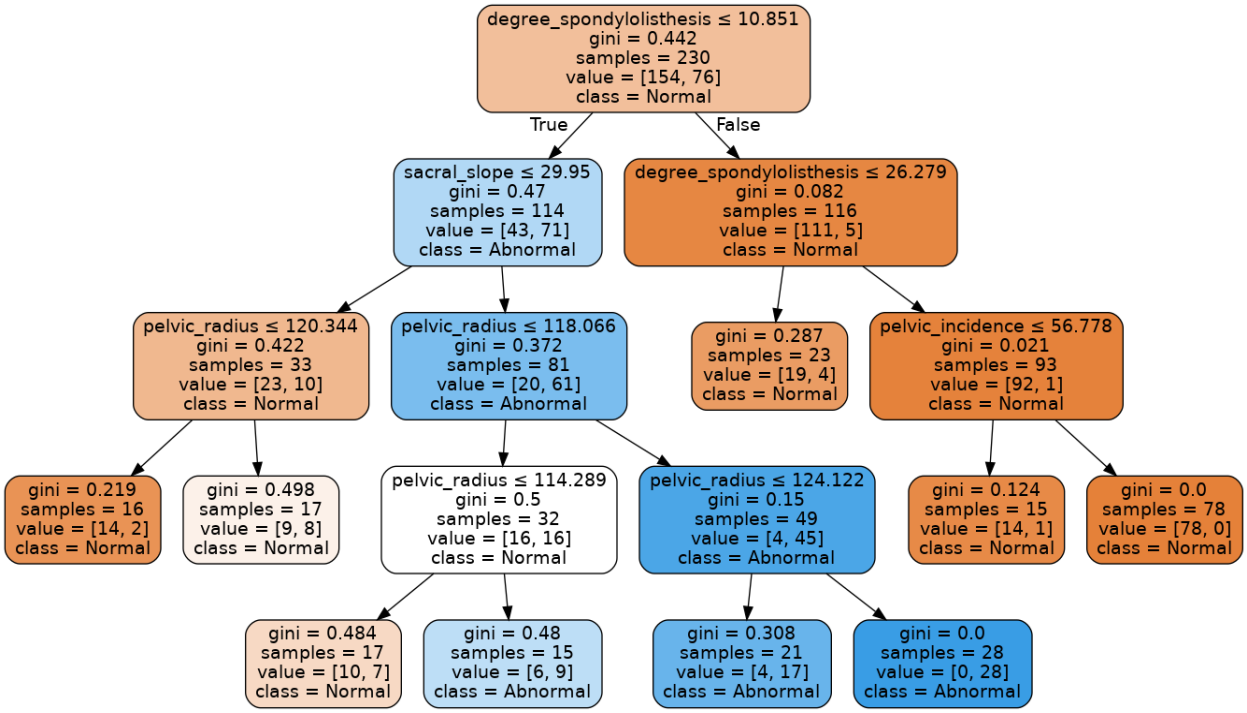


Fig 1(b) : Minimum number of leaf nodes = 15

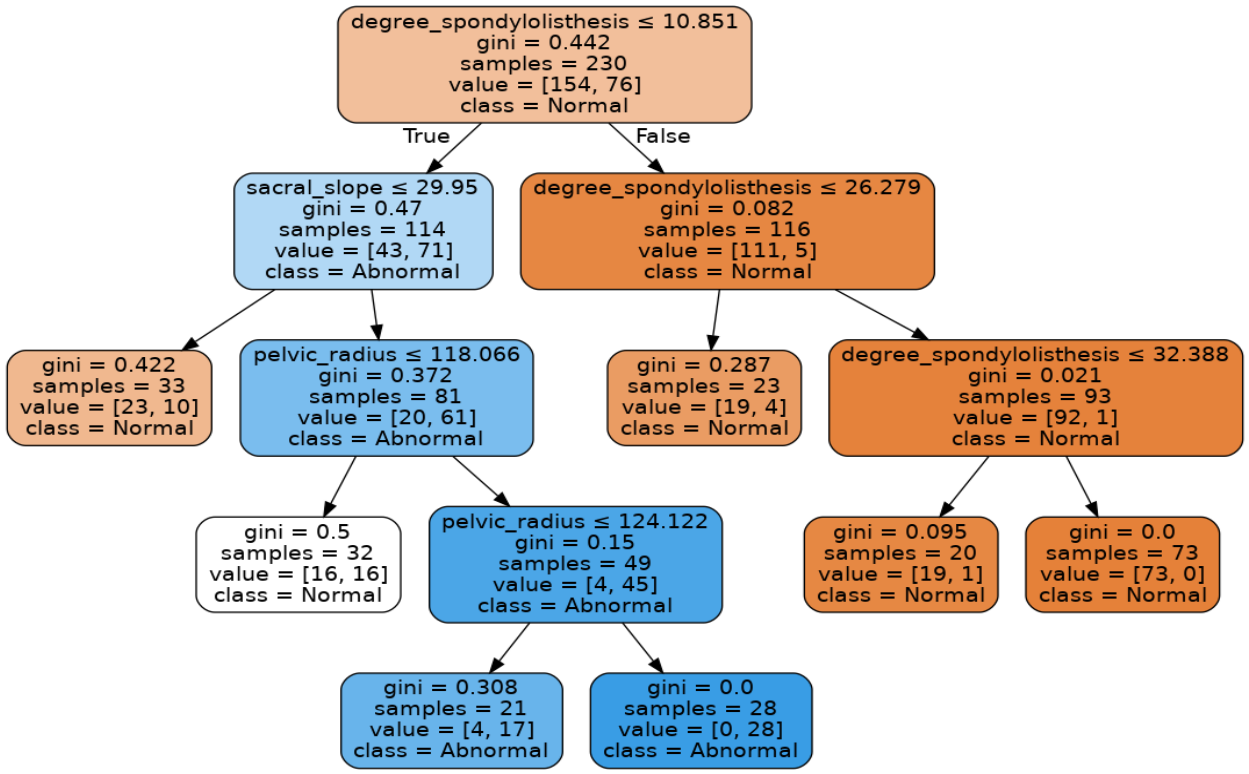


Fig 1(c) : Minimum number of leaf nodes = 25

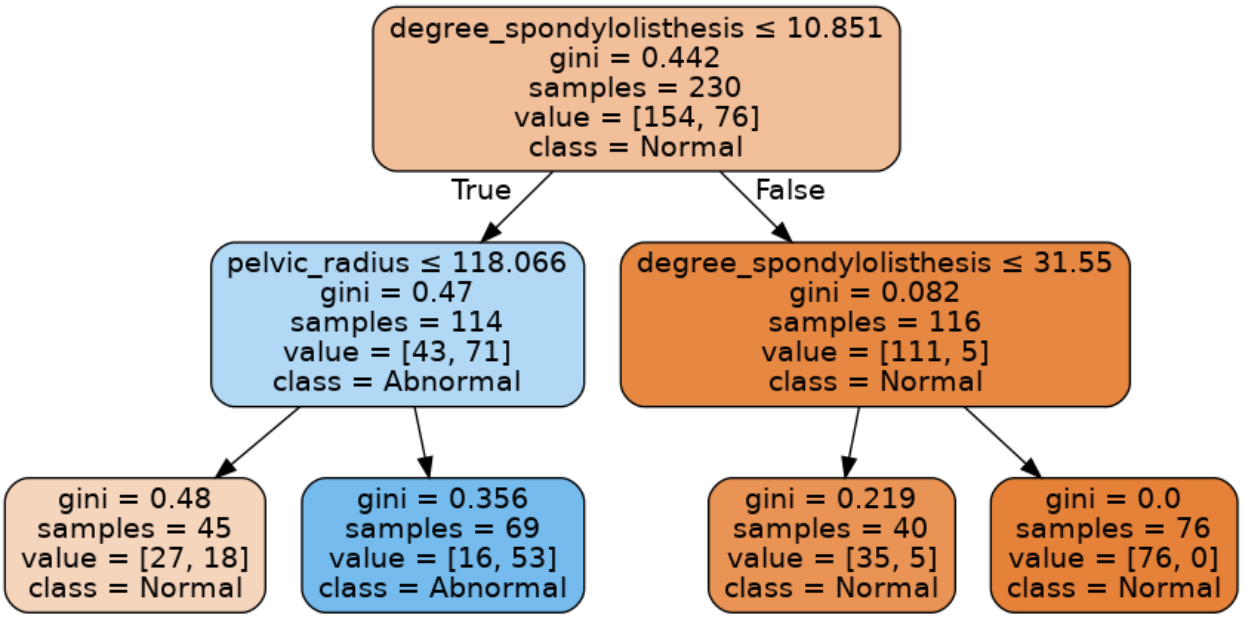


Fig 1(d) : Minimum number of leaf nodes = 40

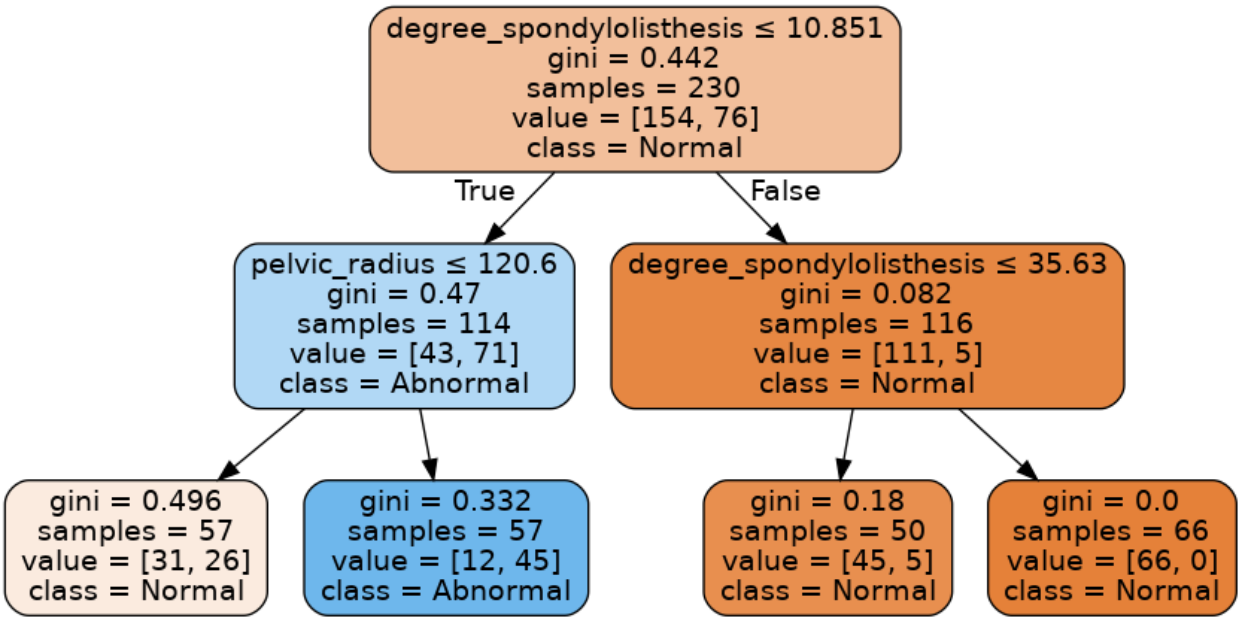


Fig 1 (e) : Minimum number of leaf nodes = 50

Why is there a difference in the 5 trees when the minimum number of leaf nodes are of varying numbers?

The first difference that is evident is that the number of leaf samples at 5 is much bigger and deeper. Since each leaf node has to contain at least 5 records, which is not a great restriction, the model that is built tries to fit in all training data points.

In other words, a smaller leaf node value makes the model **more prone to overfitting** and **capturing noise** in the data. This is the reason that the tree has a high depth and is more complex than the others.

With increasing number of minimum leaf node restrictions, the tree grows to be smaller in depth. However, if the depth is too low, there is bound to be errors and the danger of underfitting at the cost of being less computationally expensive.

We can observe that **the optimum minimum of leaf nodes** could be 15 in this case.
Hence, I would prefer the decision tree from figure 1 (b) as there is a tradeoff between underfitting, overfitting , being computationally expensive and having a complex hypothesis.

1b) **Plot of Metrics**

	Accuracy	Precision_Normal	Precision_Abnormal	Recall_Normal	Recall_Abnormal
5	0.7500	0.807692	0.642857	0.807692	0.642857
15	0.8000	0.846154	0.714286	0.846154	0.714286
25	0.7750	0.783333	0.750000	0.903846	0.535714
40	0.7625	0.811321	0.666667	0.826923	0.642857
50	0.7750	0.814815	0.692308	0.846154	0.642857

Accuracy : Accuracy is defined as TP + TN / TP + FP + FN + TN. Intuitively, this gives us the ratio of correctly classified records to the sum of records observed. As observed, the accuracy value is highest at the min_leaf 15. The trend is sinusoidal ; it increases and then, decreases. This happens because as the min_leaf value increases, the tree becomes less complex with lower depth than its previous value. But, once it becomes simple beyond a threshold, it drops again due to impurity. Here, the preferred decision tree is yielding a 0.8125 accuracy.

Precision: Precision is a metric defined as TP/TP+FP. It is the ratio of correctly identified positive records to the sum of all records that were identified positive. Intuitively, how precisely it predicted the positive examples. 88% of the positive records were identified by the preferred decision tree here.

Recall: Recall is a metric that is defined as TP/TP+FN. It is the ratio of correctly identified positive examples to the sum of actual positive examples.

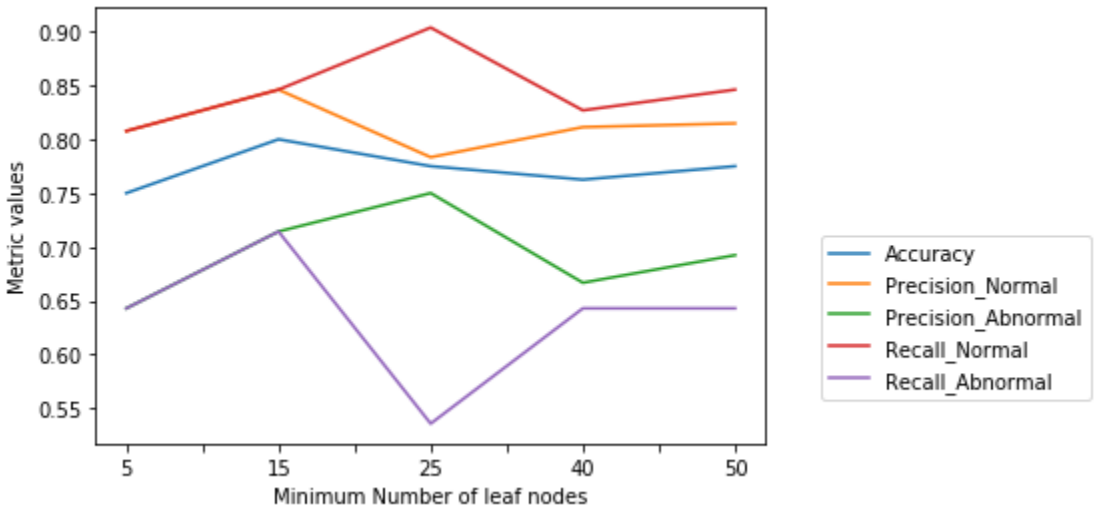
We observe in here that both the precision and recall metrics also have a phase where they increase, reach optimum and then reduce, just like accuracy.

However, since this is medical data, it is important to capture positive examples correctly. In other words, false negatives should be nearly zero. That is, the number of 'abnormal' people categorized as 'normal' should be as minimum as possible.

The preferred decision tree yields a 0.75 value. 75% of abnormal records were identified so.

Hence, we would have to choose a decision tree that has a high value of Recall_Abnormal. That happens at min_leaf = 15. It also has a good accuracy and fairly good values of the other metrics.

Preferred tree = min_leaf 15.



QUESTION-2:

2a)

In this dataset, we are dealing with three target classes : {Normal, Hernia, Spondylolisthesis}.The decision trees are as follows. It follows the same trend of being complicated for min_leaf 5 and goes on to reach an optimum (which we need to discover), and then becomes simple (but, too simple? Metrics will answer).

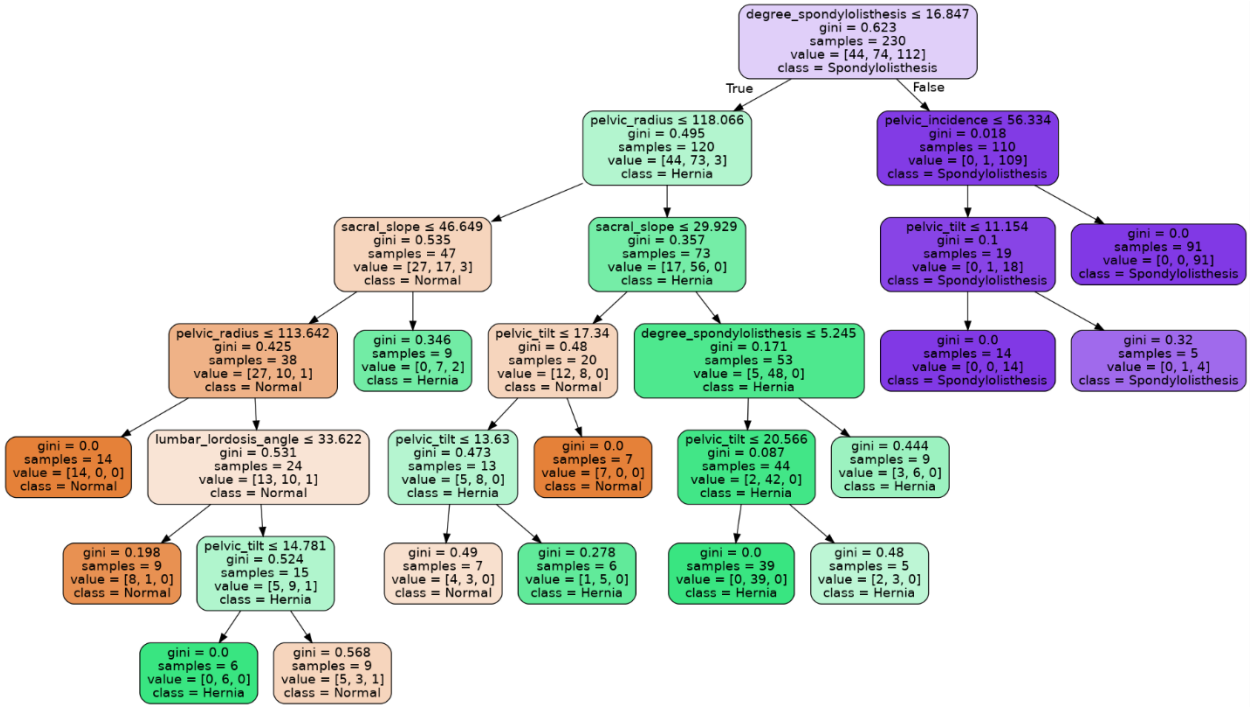


fig2(a) : Minimum number of leaf nodes = 5

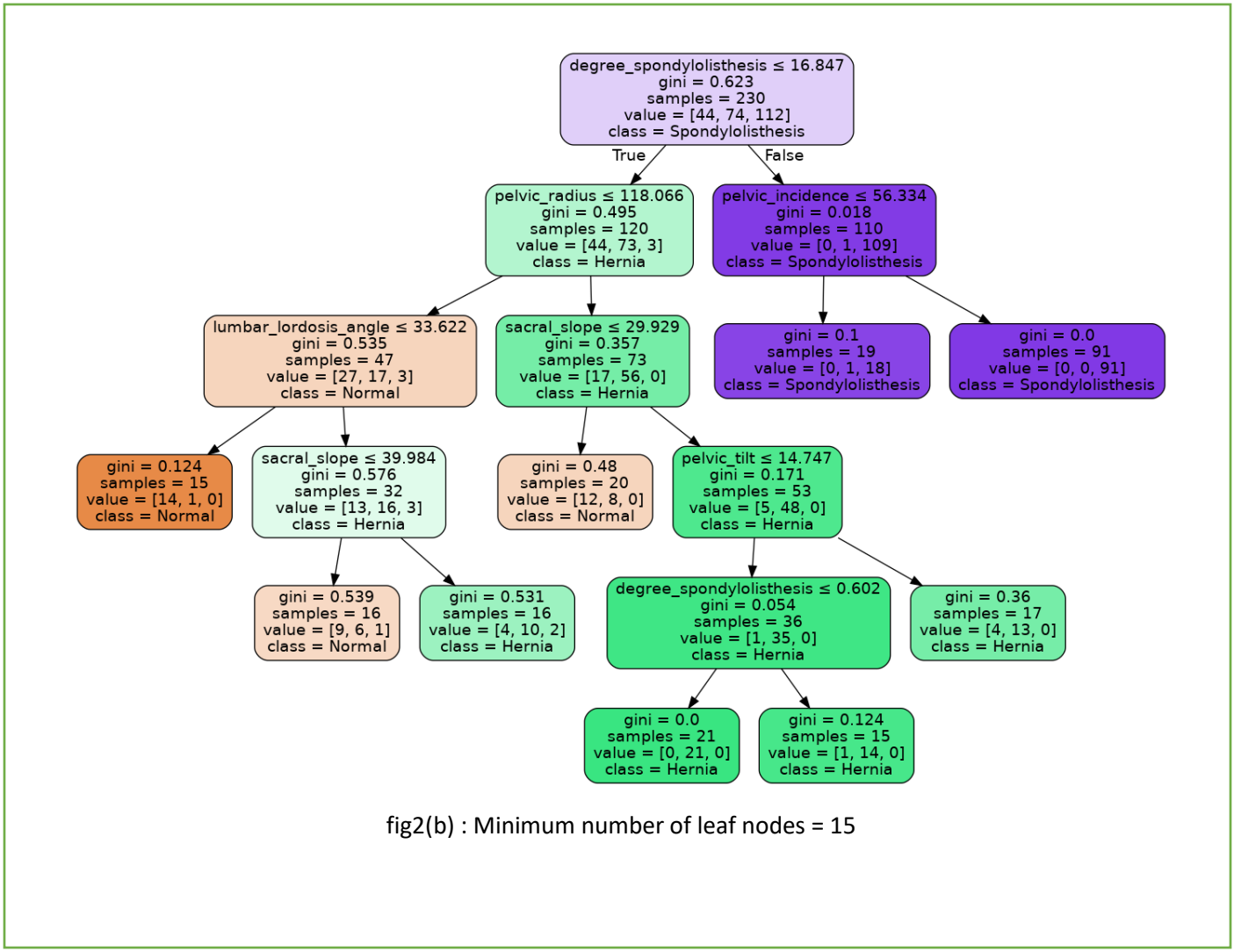


fig2(b) : Minimum number of leaf nodes = 15

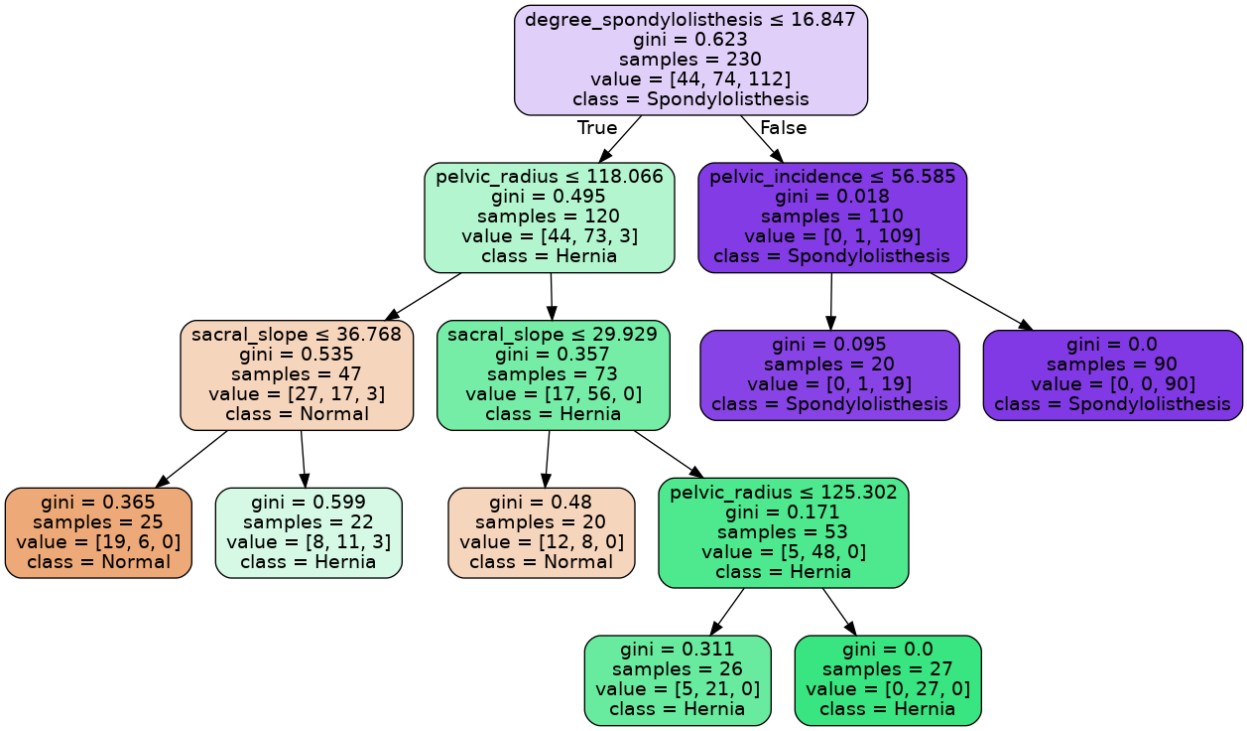


fig2(a) : Minimum number of leaf nodes = 25

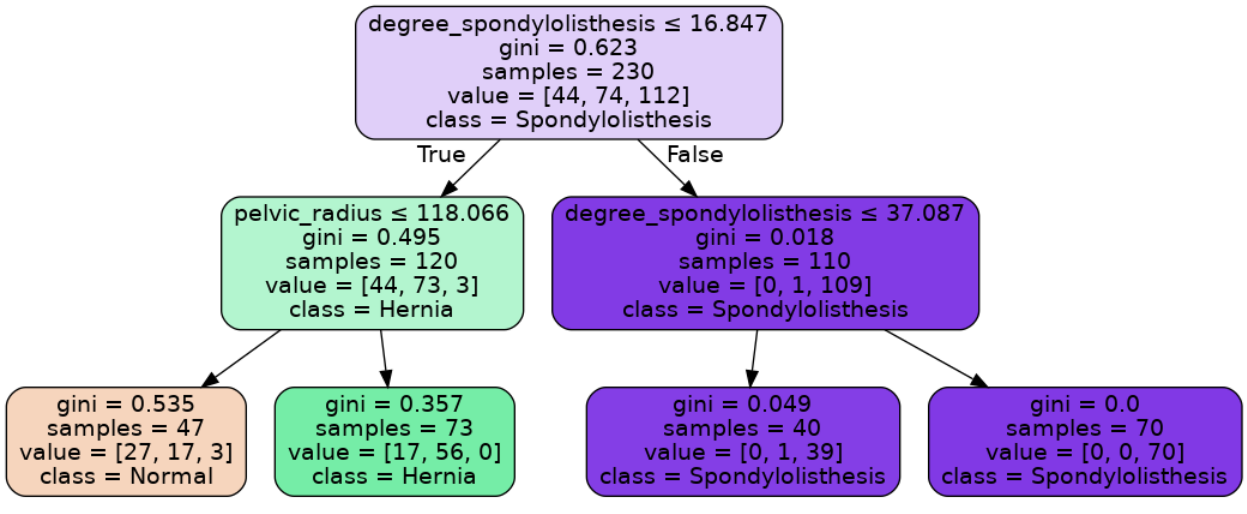


fig2(a) : Minimum number of leaf nodes = 40

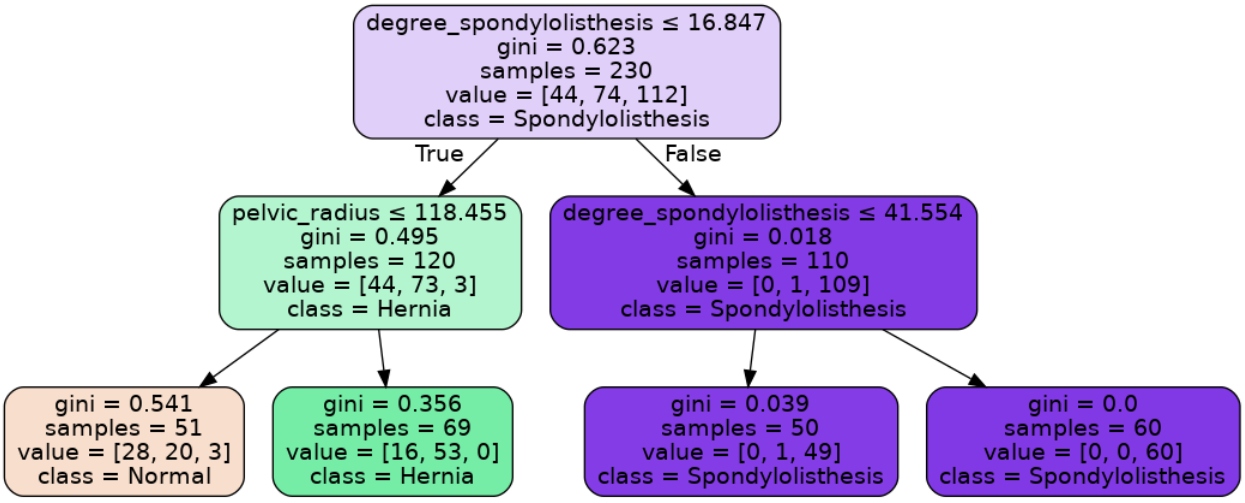
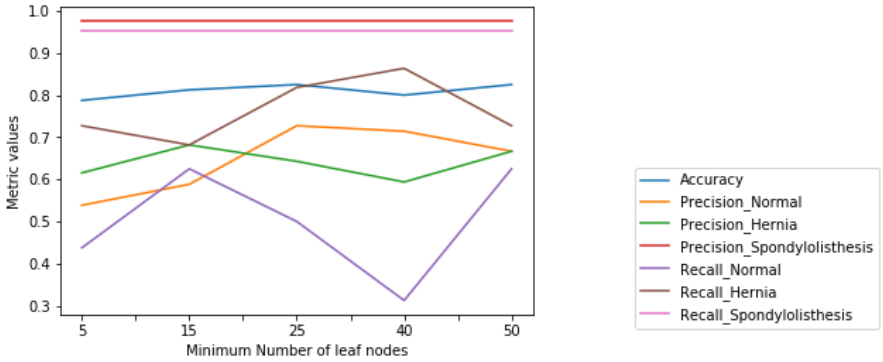


fig2(a) : Minimum number of leaf nodes = 50

How pure/right are the decision trees?

	Accuracy	Precision_Normal	Precision_Hernia	Precision_Spondylolisthesis	Recall_Normal	Recall_Hernia	Recall_Spondylolisthesis
5	0.7875	0.538462	0.615385	0.97561	0.4375	0.727273	0.952381
15	0.8125	0.588235	0.681818	0.97561	0.6250	0.681818	0.952381
25	0.8250	0.727273	0.642857	0.97561	0.5000	0.818182	0.952381
40	0.8000	0.714286	0.593750	0.97561	0.3125	0.863636	0.952381
50	0.8250	0.666667	0.666667	0.97561	0.6250	0.727273	0.952381



We see that the highest accuracy is occurring at min_leaf = 5 and it has a high recall value for abnormal states (Hernia and Spondylolisthesis).

But, considering the depth of the tree, computational expense and complexity, there is an uncertainty with validation tests.

Our test set contained only 30 records. I would prefer this tree only after reviewing metrics of other testing data as I am concerned about overfitting.

In addition, the dataset in imbalanced with the classes. More than half of the target class belongs to

"Spondylolisthesis", making it easier for any classifier to categorize data in this class correctly. Hence, I wouldn't give precision/recall of this target class as much value as the others.

I am more inclined to base my preference over the metrics of the class "Hernia" have performed as there are only 60 records belonging to this target class. The overlapping max values for this class happen to be at min_leaf 25.

As mentioned before, for medical data, it is imperative to infer recall values to choose the best model. Hence, I would prefer min_leaf as 25 as with 5 I'm uncertain about overfitting and above 25, it definitely underfits and performs badly as we see from the metrics.

Recall_Hernia = 0.81 , Recall_ Spondylolisthesis = 0.95 , Accuracy = 0.82. All other metrics for different classes also seem to be sufficiently good to be choosing this decision tree.

Preference : Min_leaf = 25.

Decision Trees of Data2 vs Decision Tress of Data3

a) The depth of min_leaf_5 of Data2 = 8 and the depth of min_leaf_5 of Data 3 = 6. This is so because of the increase in number of classes and a combination of attributes easily resulting in a leaf node. In other words, purity has increased or the ease in identifying 'Spondylolisthesis' has gone up because of the sheer volume of the number of target classes belonging to that group.

b) To identify 'Hernia', the longest depth would be 8 in min_depth = 5. In all decision trees, Hernia is difficult to reach, or it happens to be the mostly likely candidate for misclassification. This is because only 60 Hernia records exist in total. After Train-test split, the numbers would have gotten worse.

c) In Data2, the worst recall value was 62.5. However, in Data3, it is 58. Again, it is easier to classify Hernia and Spondylolisthesis as 'abnormal' than to classify abnormal into the former two because

1) Training volume
2) Imbalanced data of target class.

Hence, we observe that Gini_index of Data2 is much better (purer) than Gini_index of Data3.

Metrics of Data2 vs Metrics of Data3

a) Precision and recall of 'Spondylolisthesis' is much better than 'Hernia' due to the training volume.
Recall of abnormal at the preferred min_leaf and recall of 'Hernia' at min_leaf value happens to be the same for this randomly split dataset and it could vary in each run. However, it is unlikely that recall value of Hernia would be better than the recall value of abnormal.

b) Accuracy improves from preferred min_leaf at Data2 (15) to preferred min_leaf at Data3(25). This happens because Data2 is a binary class dataset and Data3 is a multiclass dataset. When the number of classes increase, it logically follows that the training instances should also increase to tune the model well.

c) Precision and recall of 'Normal' is much better in Data2 compared to Data3. This has occurred due to node splits that follow the purity of the remaining classes as well, thus giving rise to different decision trees although logically the same records that are marked 'normal' in data2 are the same ones in Data3.

This demonstrates how the rest of the classes also have a role to play in the seemingly static records copied from a dataset.

QUESTION-3:

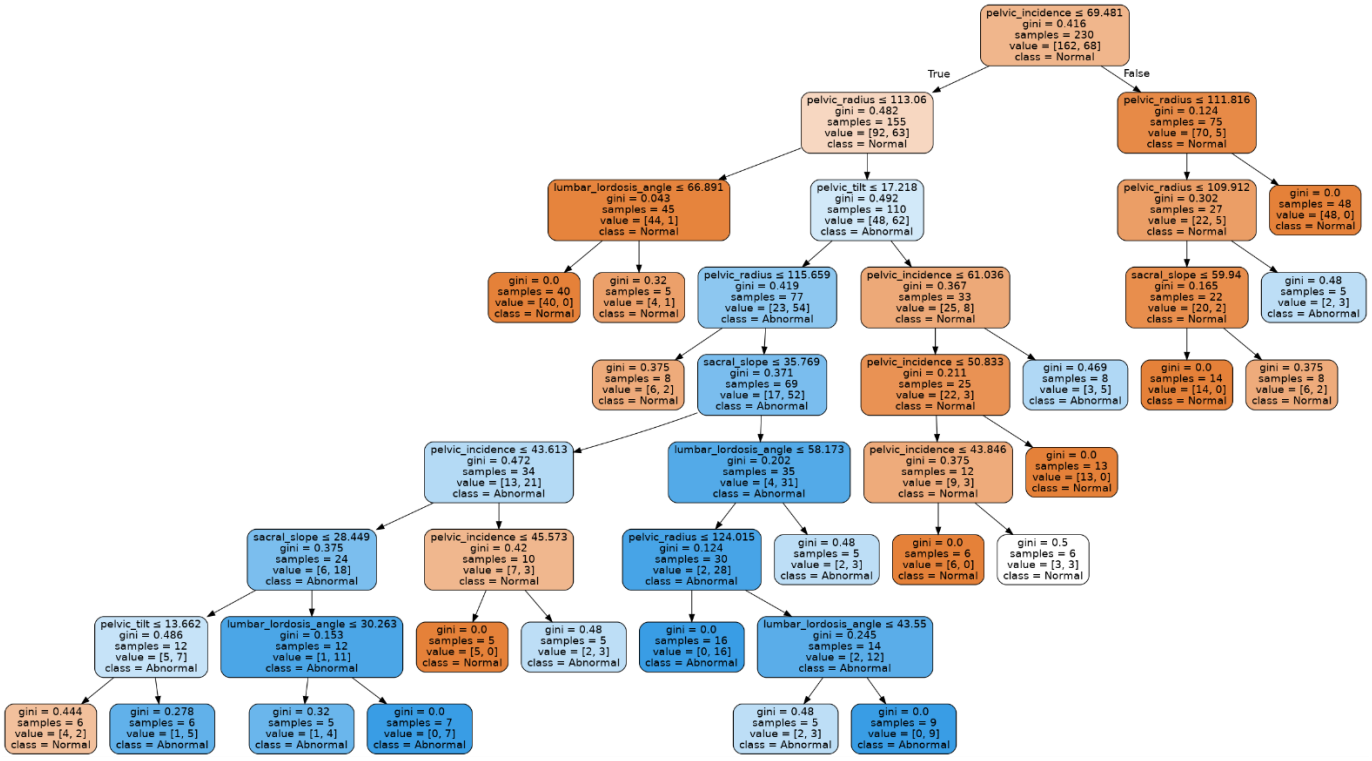


Fig3(a) : Minimum number of leaf nodes : 5

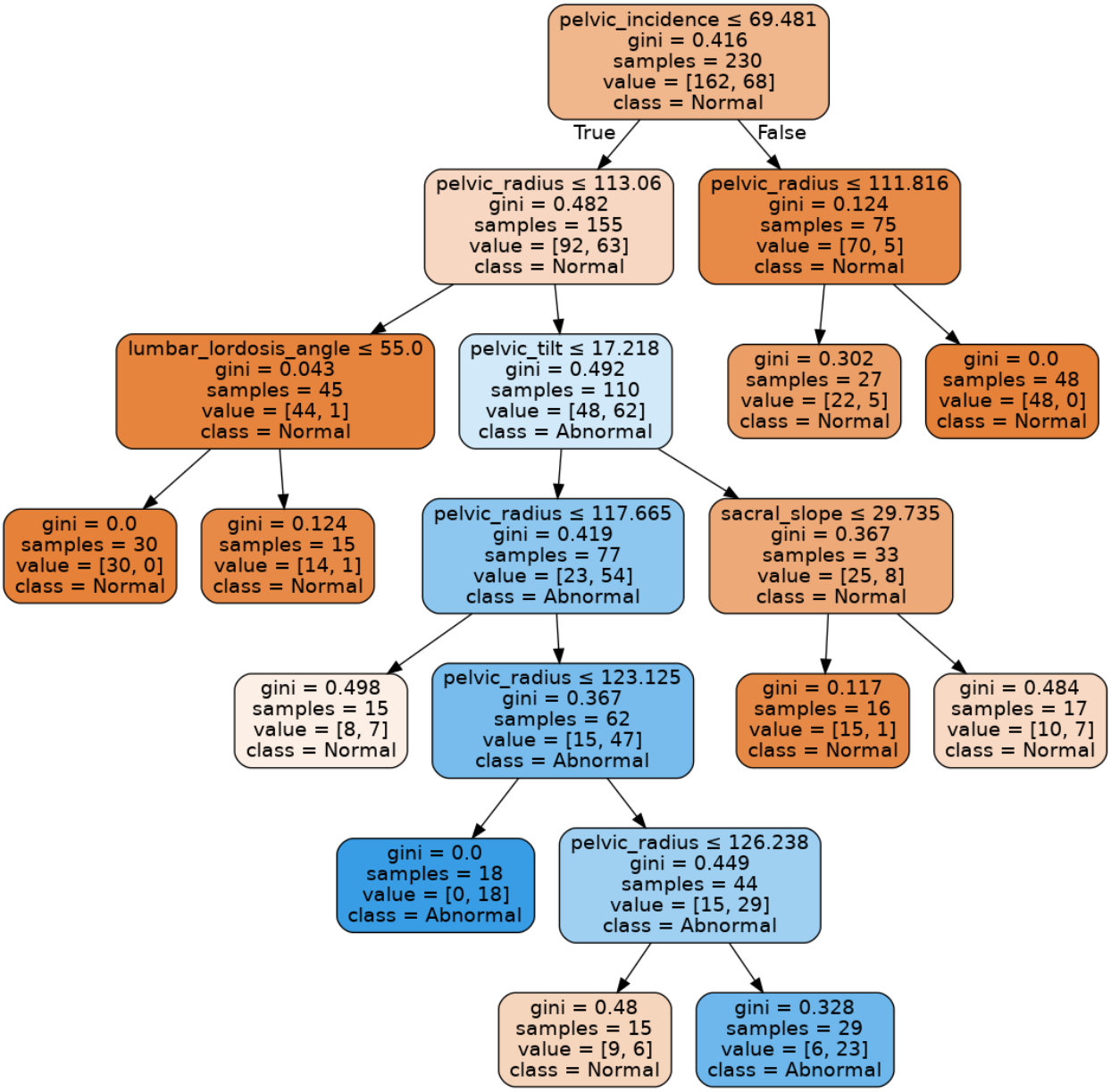


Fig3(b) : Minimum number of leaf nodes : 15

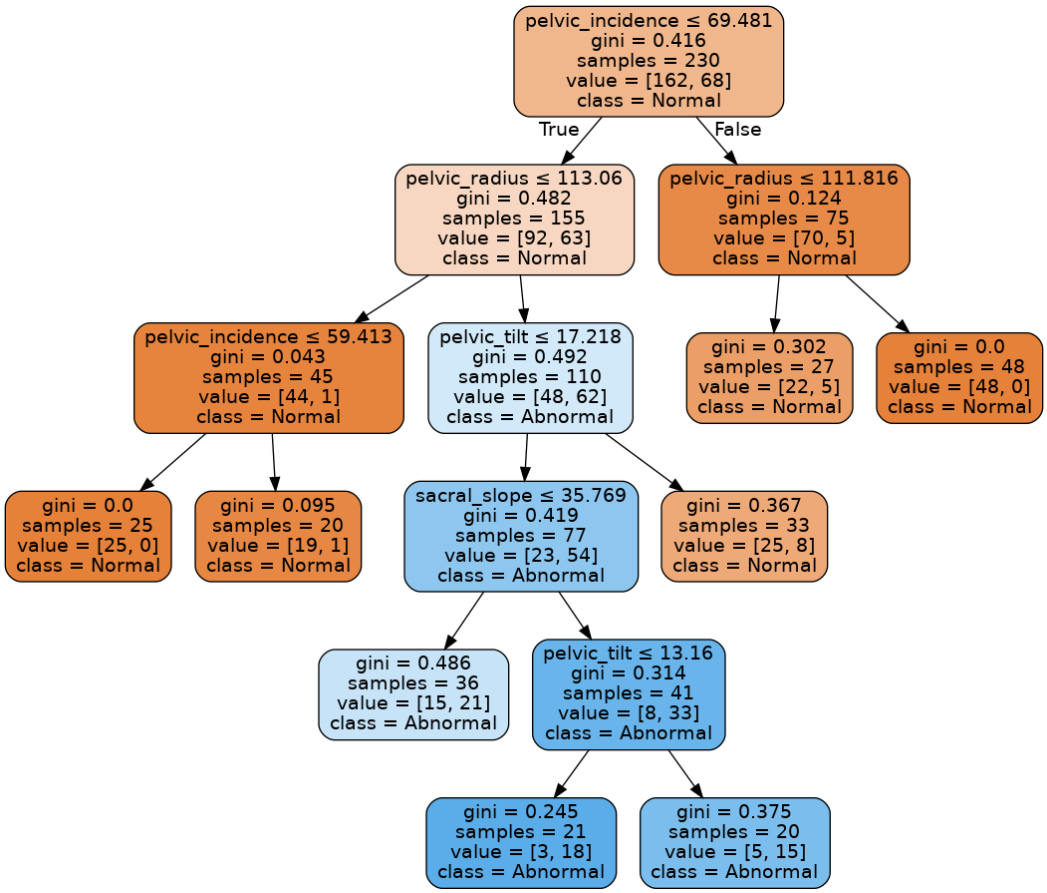


Fig3(c) : Minimum number of leaf nodes : 25

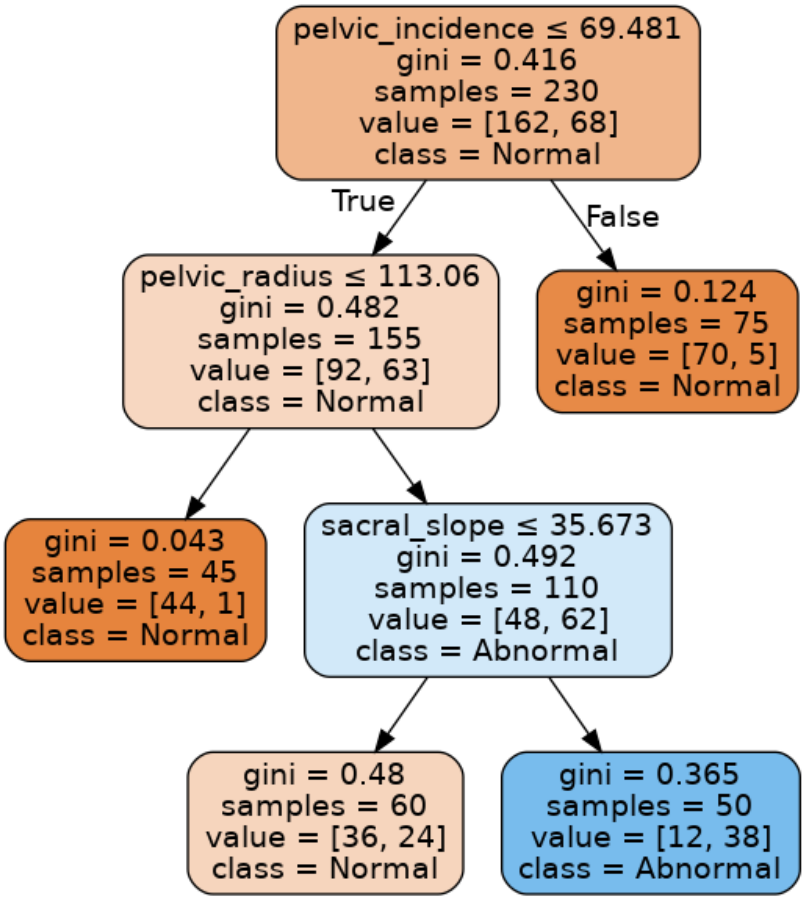


Fig3(d) : Minimum number of leaf nodes : 40

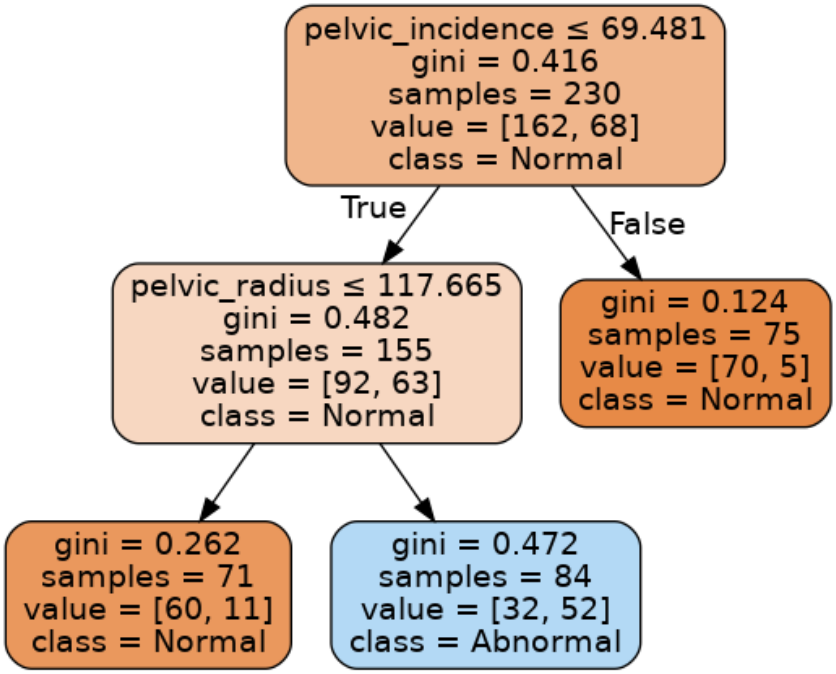


Fig3(e) : Minimum number of leaf nodes : 50

I would prefer to choose Min_leaf = 15, to avoid both overfitting and underfitting.

In this problem, we identify the highest correlated attribute and drop it from the dataset.

Also, we observe here that the depth reduces by 1 in the trees. It has made the model simpler. But, let us go through the metrics to analyze the results.

What is correlation?

Correlation intuitively means how strongly connected/related an attribute is to the target class. Observing from the previous decision trees in 1 and 2, the root node happens to be the 'degree_spondylolisthesis' attribute.
When this is chosen as the root node to split the other nodes on, this indicates that this node is the most strongly related node to the class. Then, the next most strongly connected attribute is chosen and so on.

Mathematically, these are our correlation values of attributes against the target class. To initiate calculations and arrive at a value, the classes have been categorized(binarized) from 'Normal' and 'abnormal' to 0s and 1s. As observed, our maximum value is the 'degree_spondylolisthesis' attribute.

When this column is dropped, there is an obvious feature reduction that thwarts overfitting and this makes the algorithm faster than before.

Sometimes highly correlated features can mask interactions between other features and also might have increase in speed and accuracy and reduction in computational expense. But, with certain datasets, there might be not a great deal of improvement from the previous state. But, we could retain the latter attributing to Occam's Razor. If it is simpler and has nearly the same metrics, we could rather hold on to the simpler model.

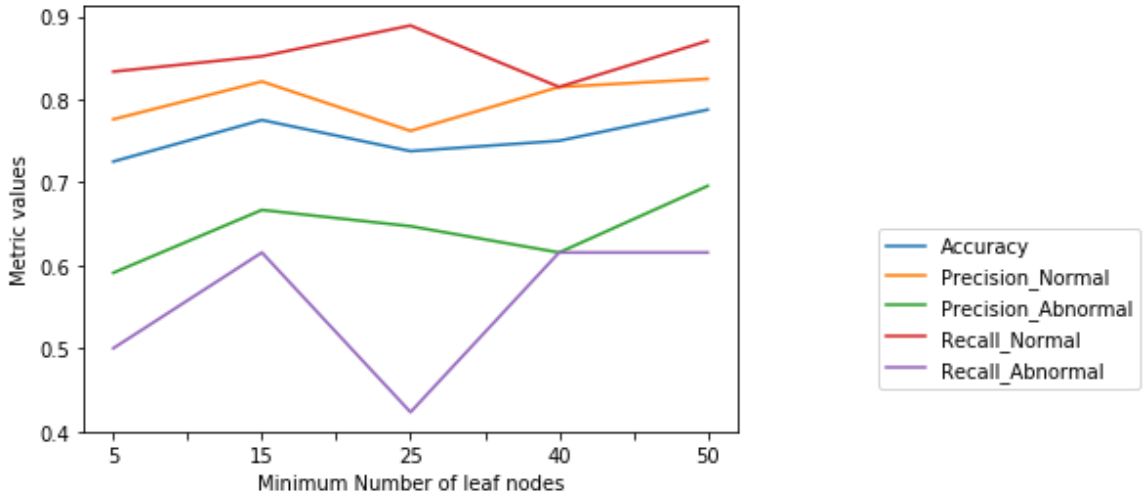
However, in this dataset and for this particular split, there is no stark improvement in the metrics; in fact, it has reduced. It seems to be strongly correlated to the target causing considerable metric contribution. This step could be treated as a wrapper algorithm process to see if the attribute is really worth dropping. Here, I do not think it is. It is better to retain the 'degree_spondylolisthesis' attribute to achieve more accuracy.

The gini_index value also seems to be increasing in value, indicating impurity and error.

Accuracy, precision and recall have drastically gone down from 0.89 (in question1) to 0.77 ,0.84 to 0.82 and 0.71 to 0.61 respectively.

This clearly isn't influencing the model positively and has brought down the metrics of the decision tree.

	Accuracy	Precision_Normal	Precision_Abnormal	Recall_Normal	Recall_Abnormal
5	0.7250	0.775862	0.590909	0.833333	0.500000
15	0.7750	0.821429	0.666667	0.851852	0.615385
25	0.7375	0.761905	0.647059	0.888889	0.423077
40	0.7500	0.814815	0.615385	0.814815	0.615385
50	0.7875	0.824561	0.695652	0.870370	0.615385




```
pelvic_incidence            0.353336
pelvic_tilt numeric         0.326063
lumbar_lordosis_angle       0.312484
sacral_slope                0.210602
pelvic_radius               0.309857
degree_spondylolisthesis    0.443687
Name: class, dtype: float64
```

****

Instructions to run the .py files:

1) Open Jupyter notebook and paste the .py script

2) Run the cell to view the results