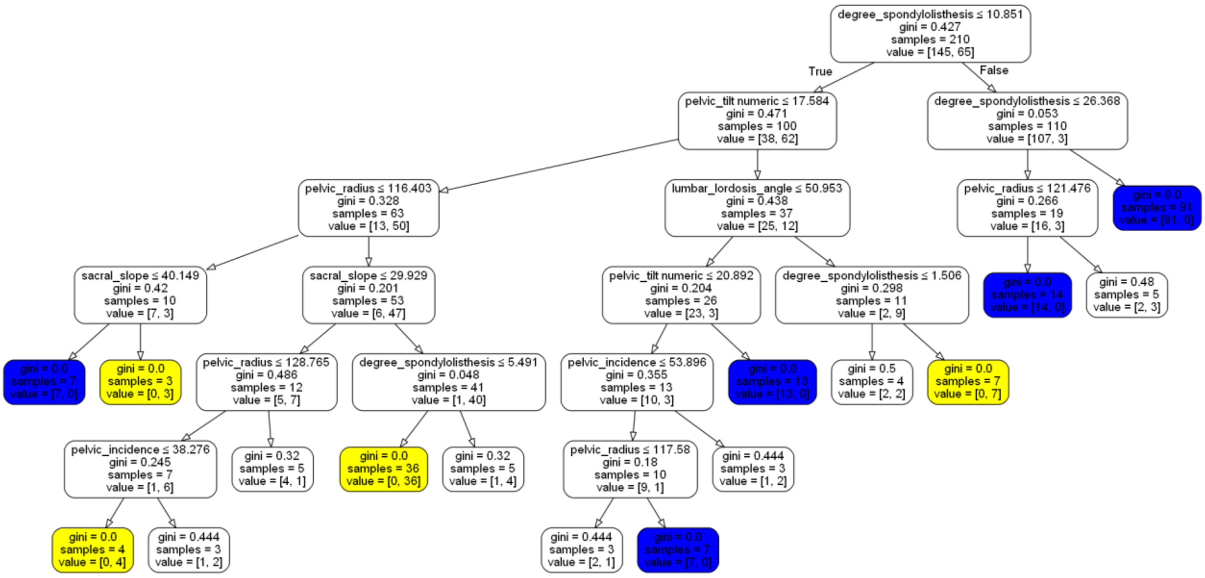
Intelligent Data Analysis – Assignment_1

Name: Piyush Sanghi   MID: M12952017

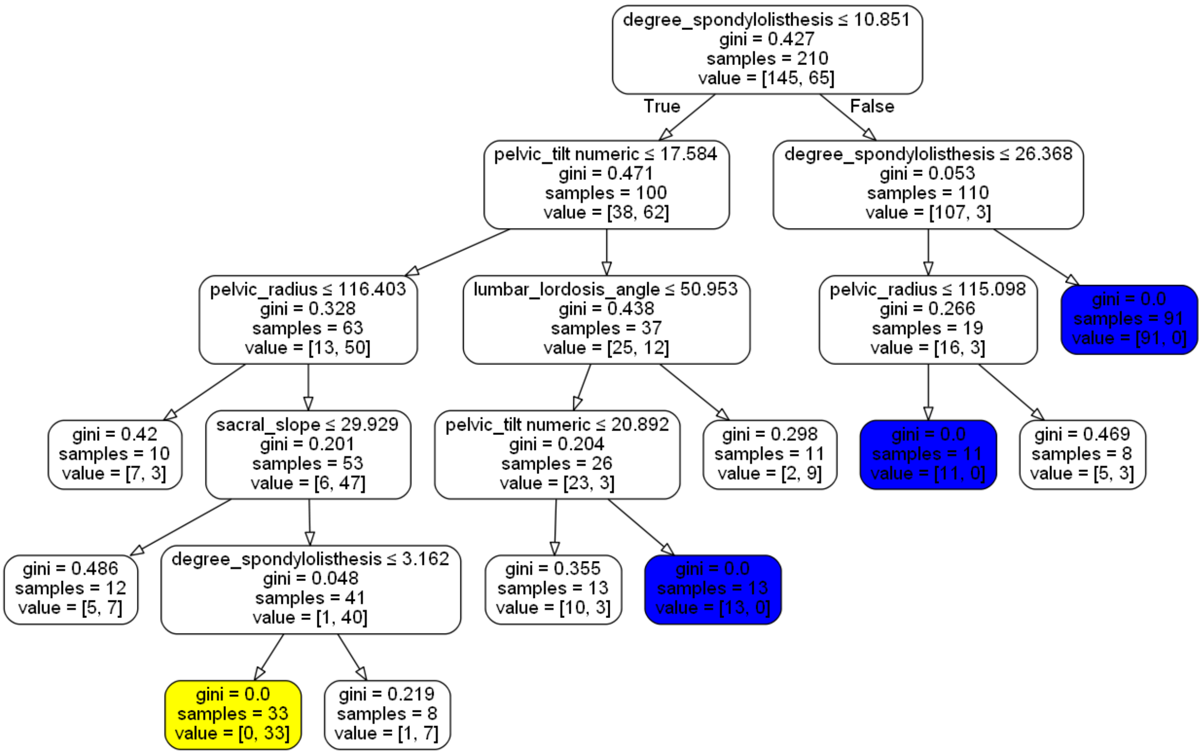
Programming language used: Python 3 (Code, explanation and output screenshots at end of the file)

Answer 1a:

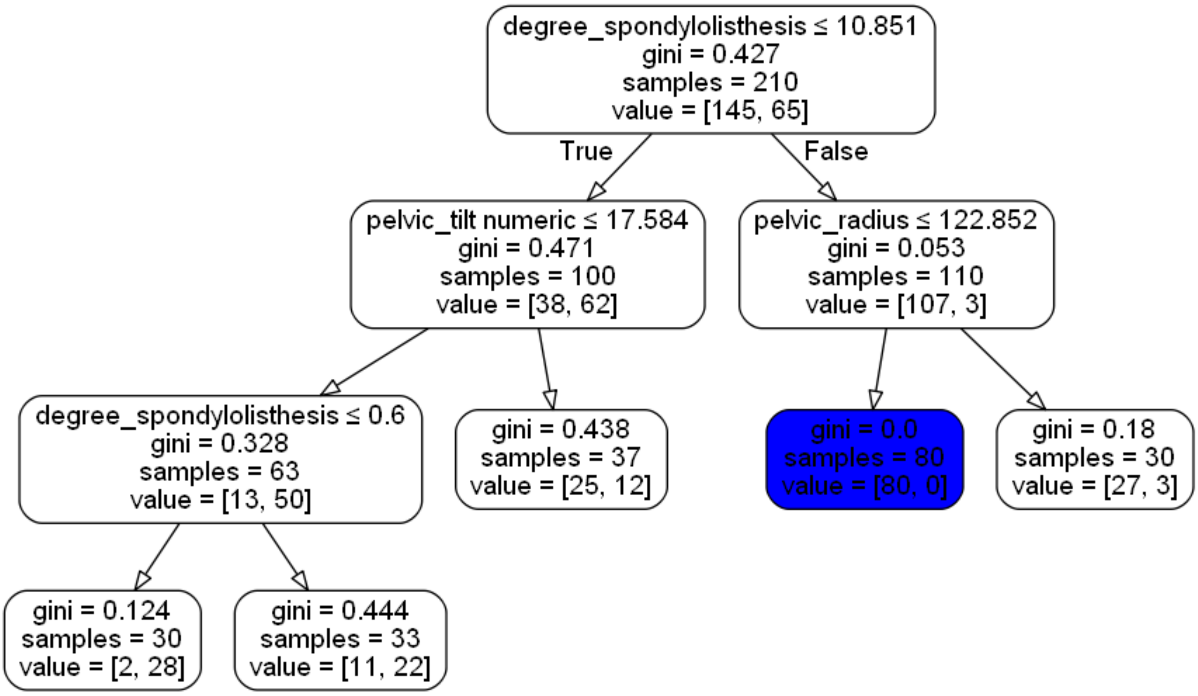
Decision Tree classifier for Minimum 3 records at leaf nodes



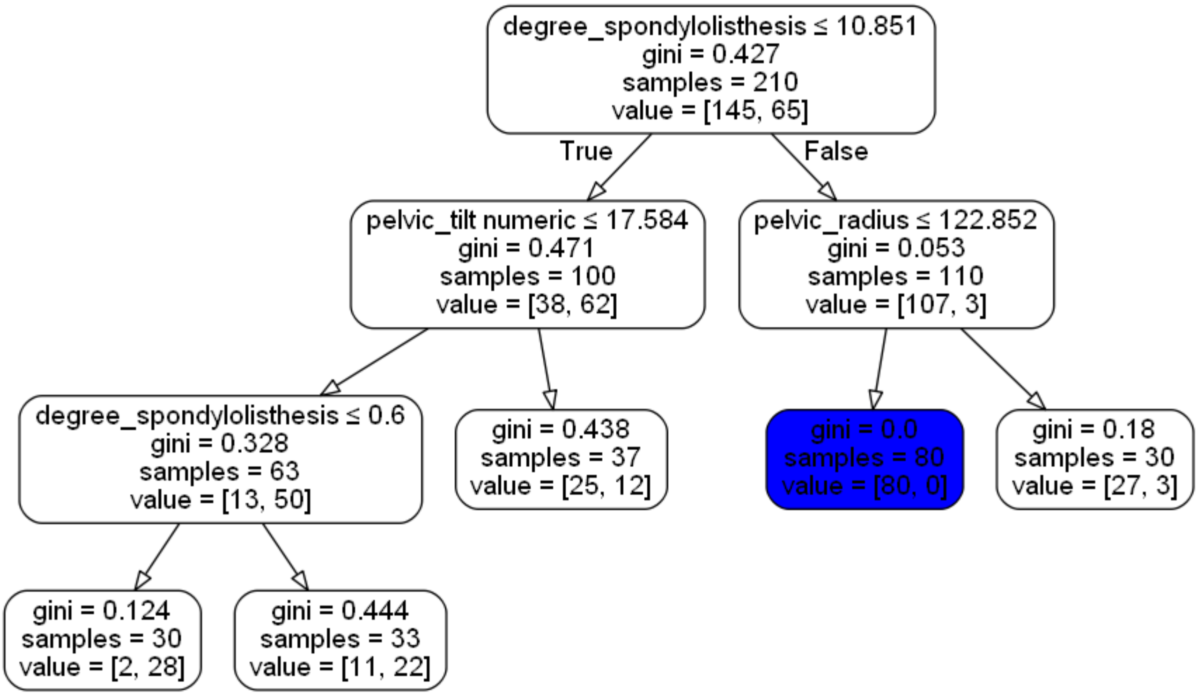
Decision Tree classifier for minimum 8 records at leaf nodes



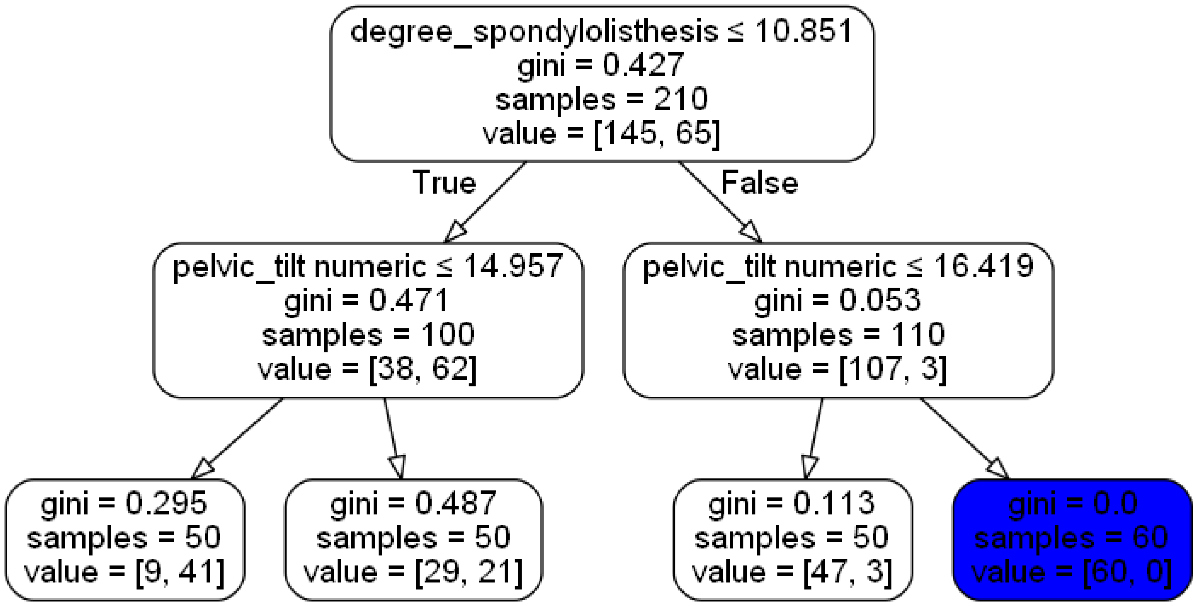
Decision Tree classifier for minimum 12 records at leaf nodes



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

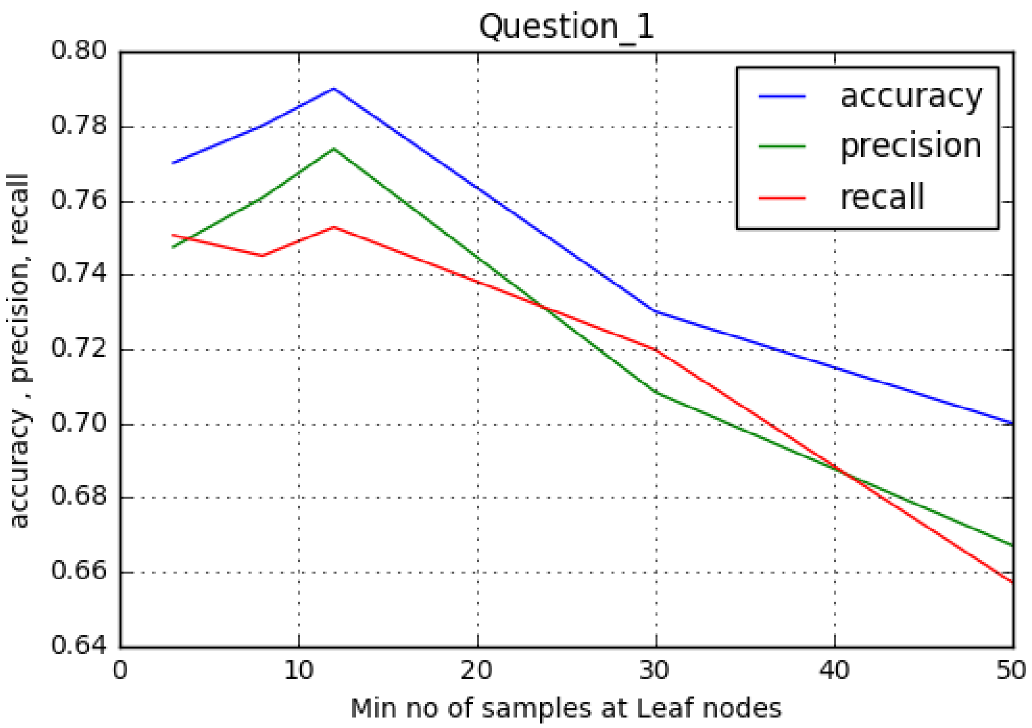
0.79, 0.7738095238095237, 0.7527472527472527

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

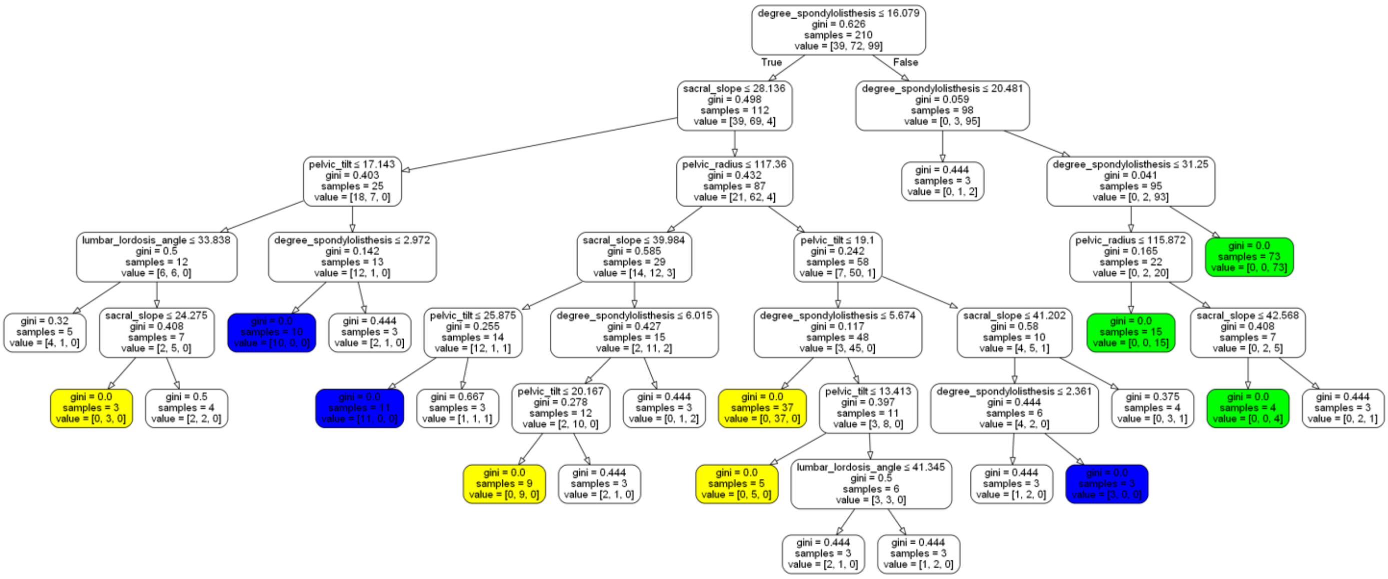


Hence by looking at the graph it is very clear that Decision tree with minimum 12 samples at leaf nodes has the best Precision, Recall and Accuracy. Hence it gives the best classification when compared to other trees.

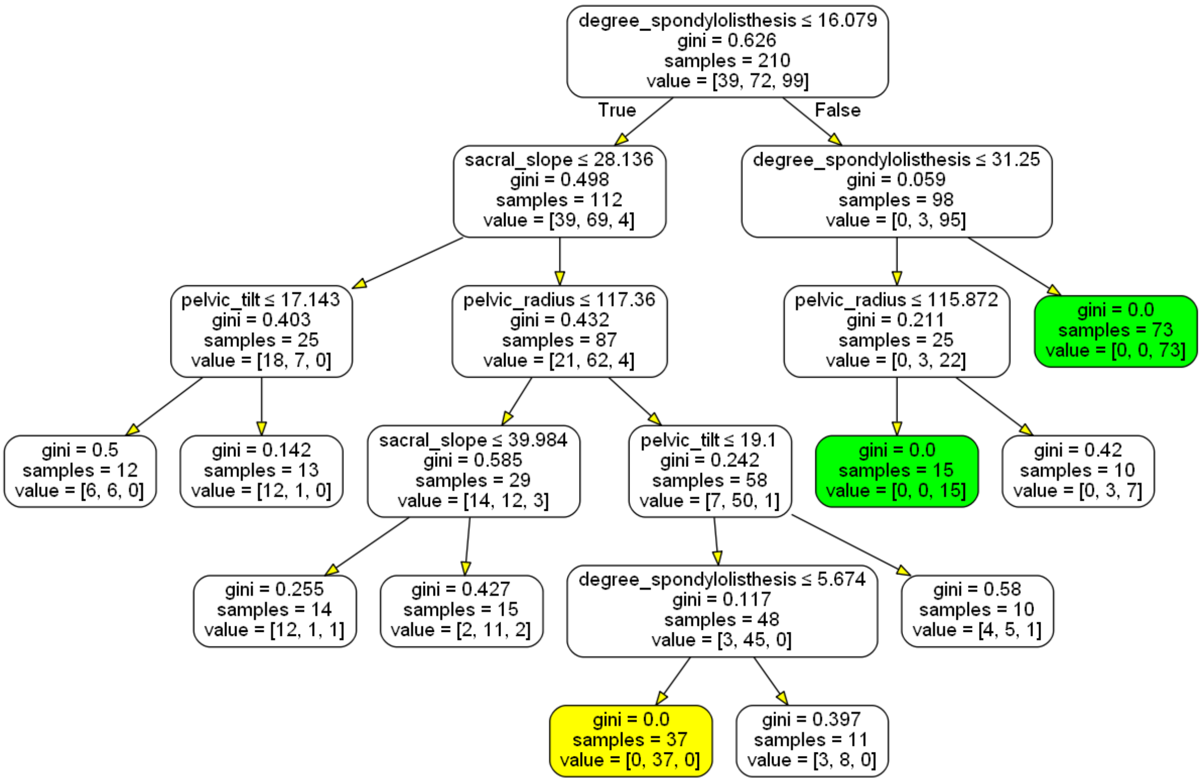
We can even notice a trend i.e. initially with increase in number of minimum samples at leaf nodes, accuracy and precision increases till 12 minimum samples at leaf nodes and then it gradually decreases. By this we can derive that initially we were overclassifying the data, hence there is less accuracy 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:

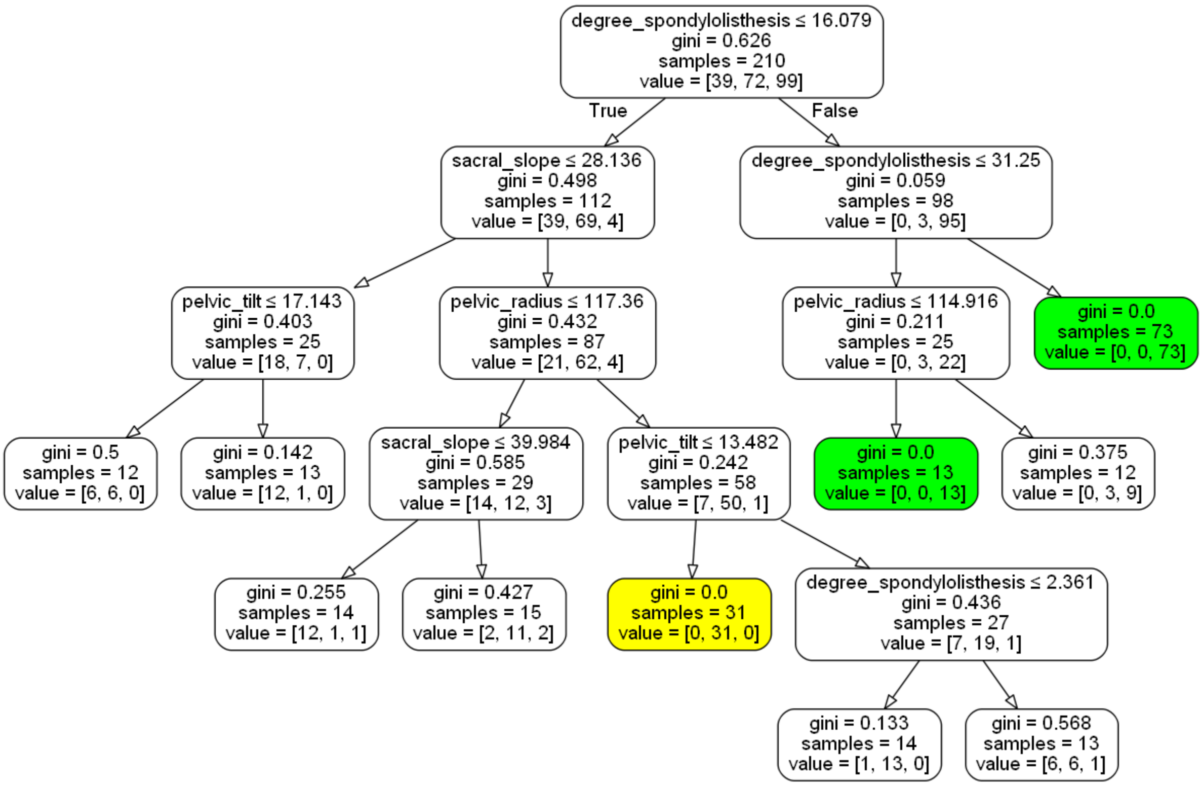
Decision Tree classifier for minimum 3 records at leaf nodes



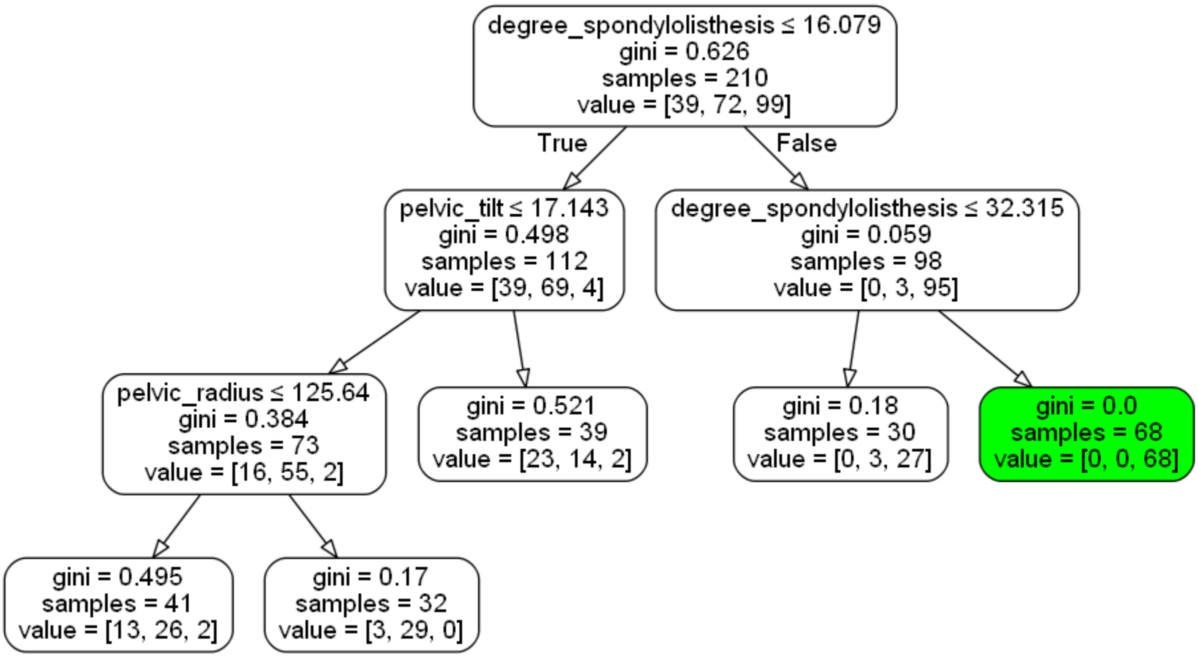
Decision Tree classifier for minimum 8 records at leaf nodes



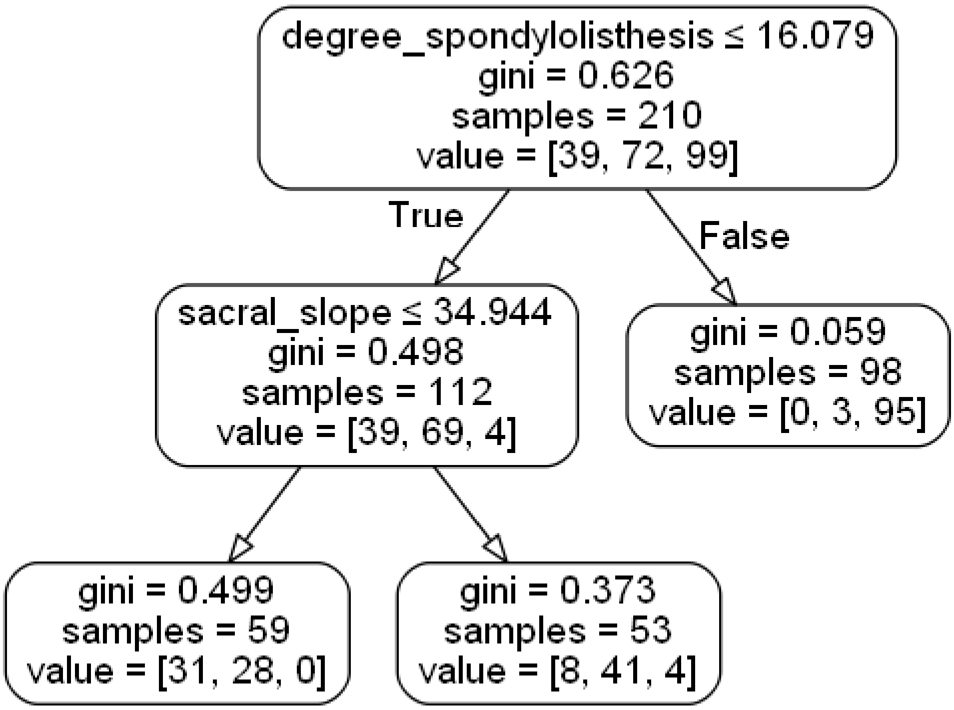
Decision Tree classifier for minimum 12 records at leaf nodes



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

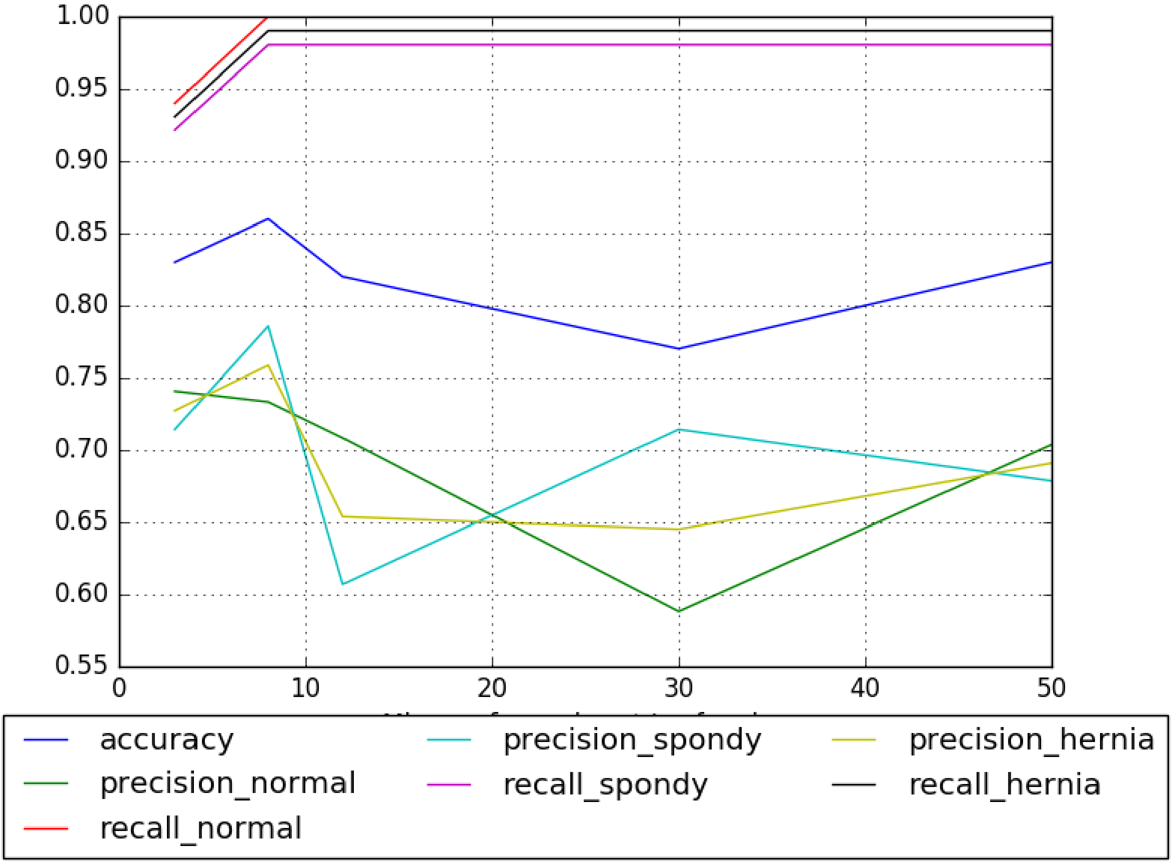
Precision, recall for class hernia is 0.6451612903225806, 0.99009900990099

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

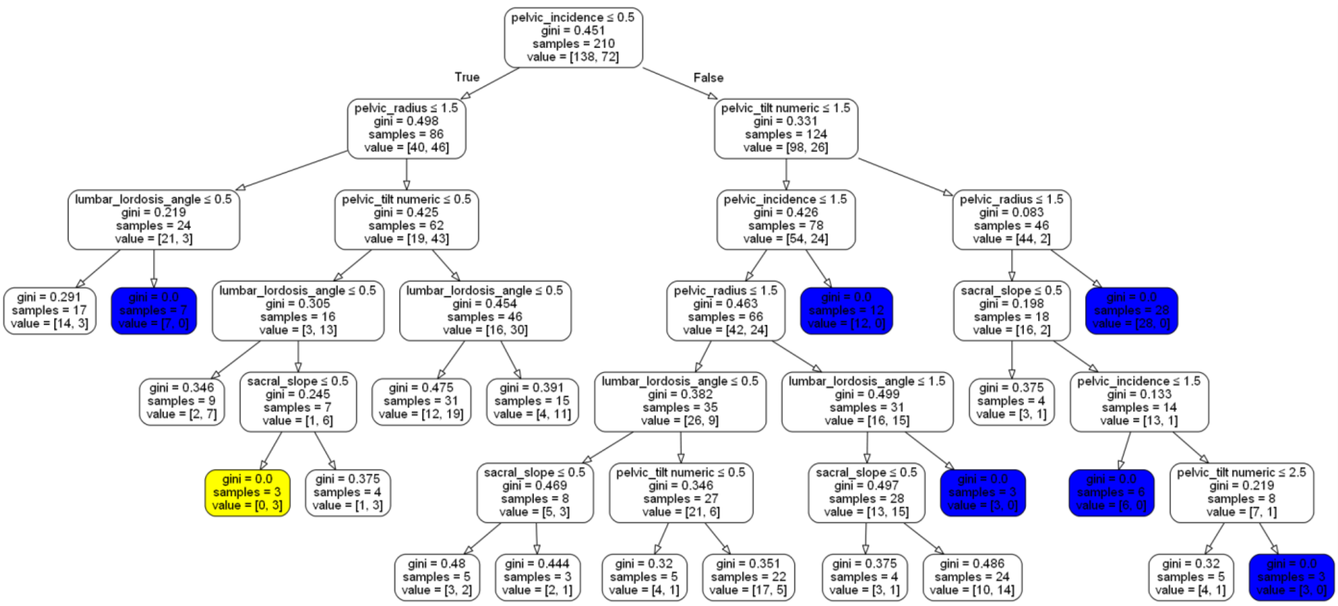
Boundaries for sacral_slope are 13.3669307, 40.38258942, 67.39824815, 94.41390687, 121.4295656

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

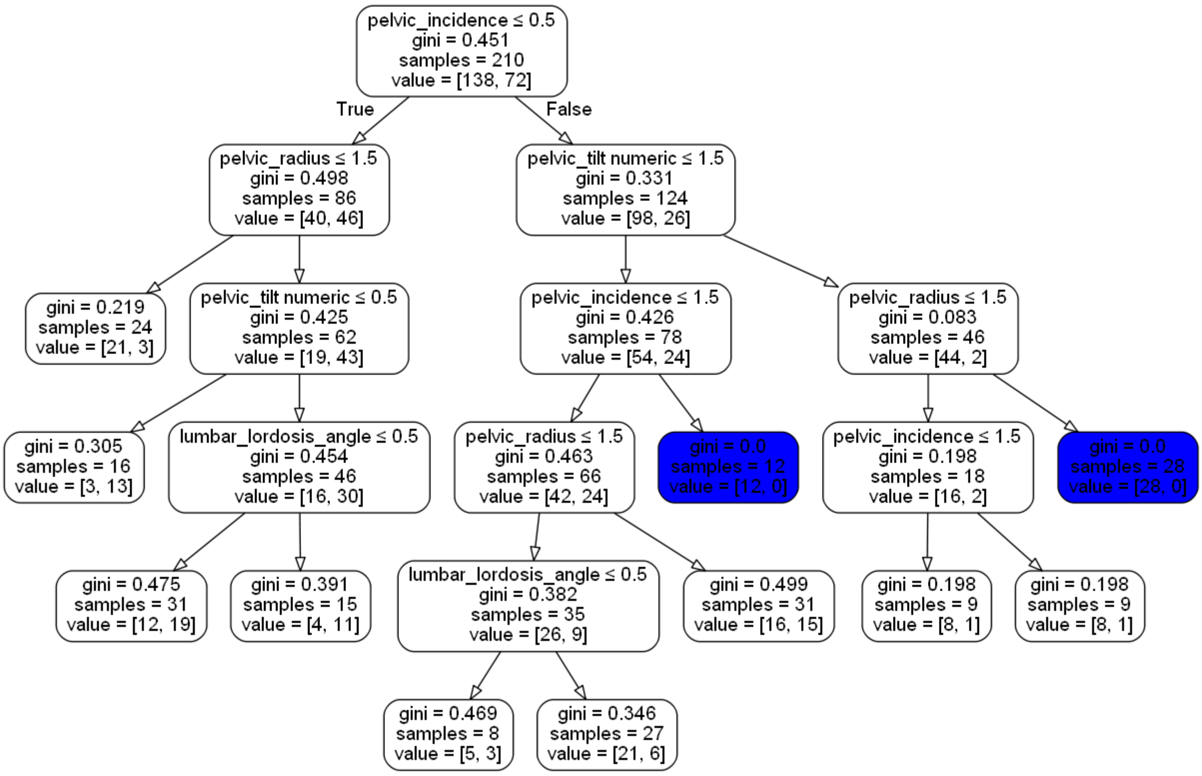
Boundaries for degree_spondylolisthesis are -11.05817866, 96.34213653, 203.74245172, 311.14276691, 418.5430821

Answer 3b:

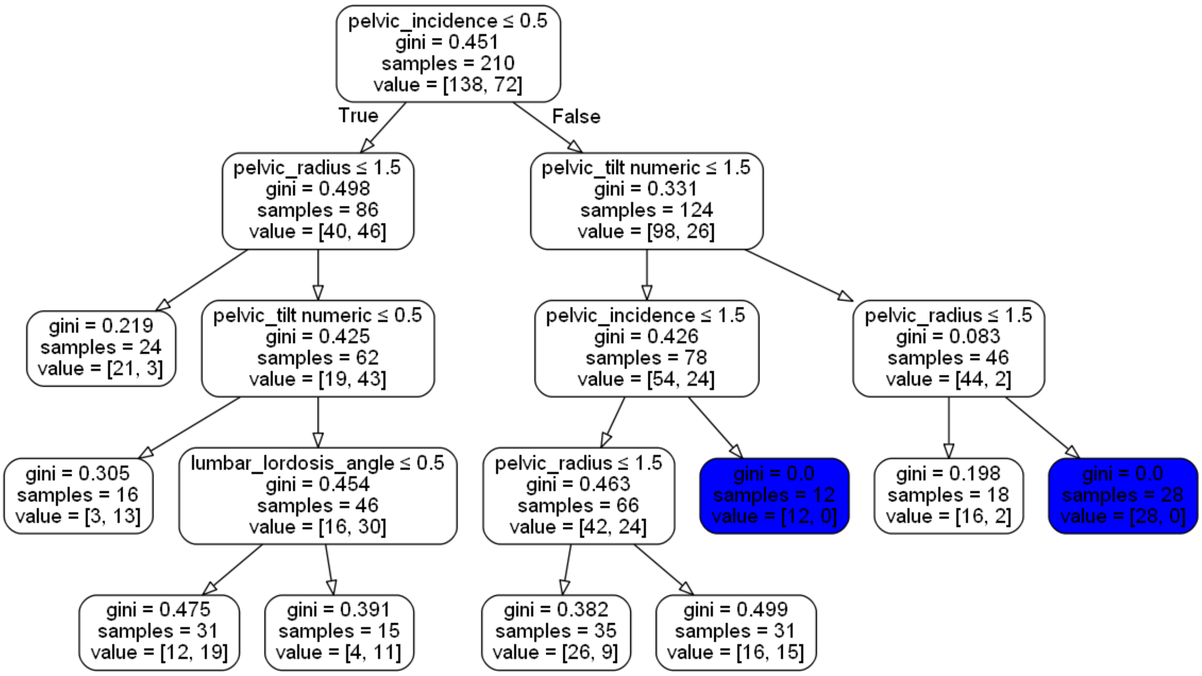
Decision Tree classifier for Minimum 3 records at leaf nodes



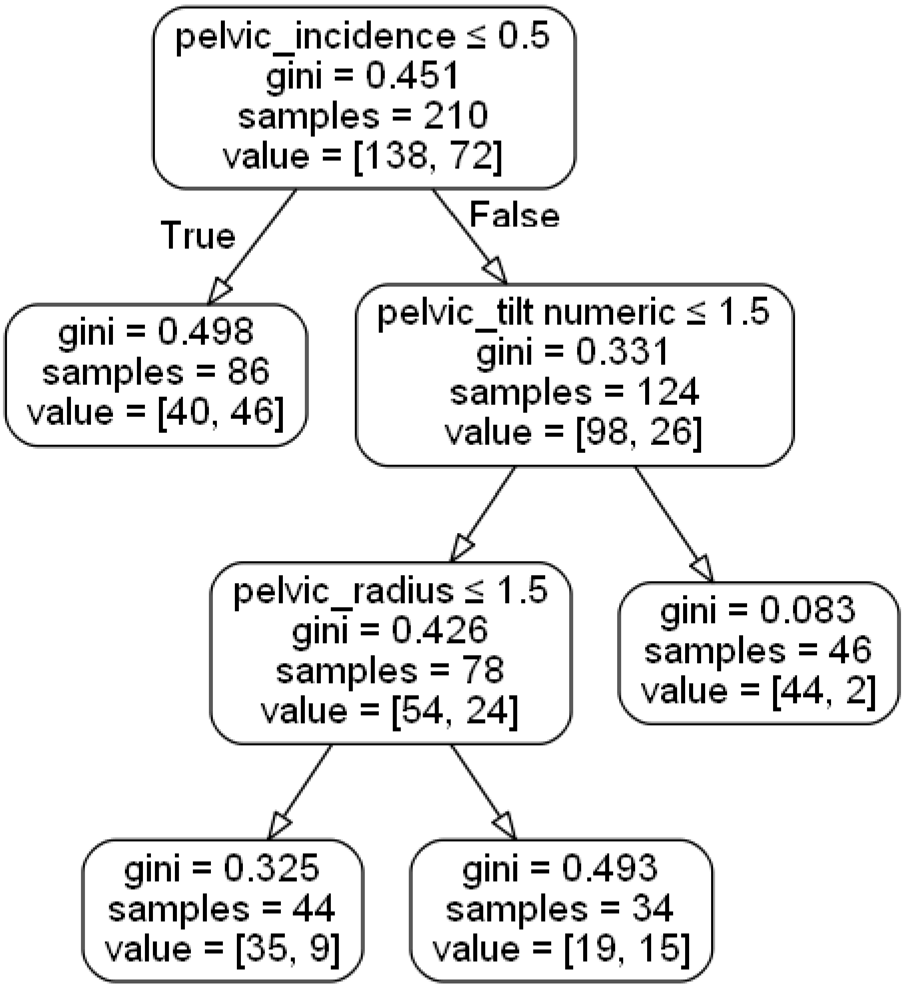
Decision Tree classifier for Minimum 8 records at leaf nodes



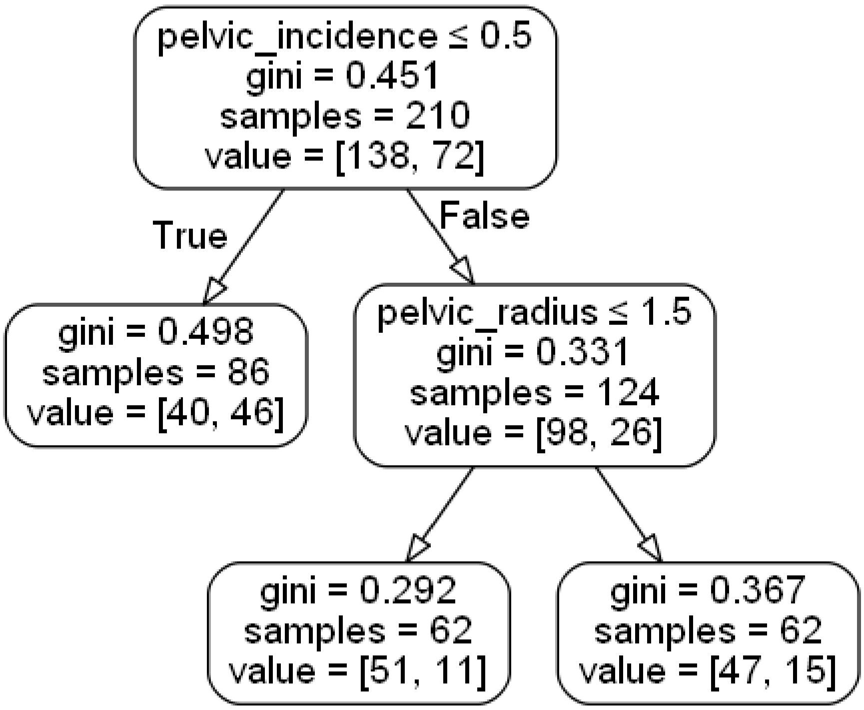
Decision Tree classifier for Minimum 12 records at leaf nodes



Decision Tree classifier for Minimum 30 records at leaf nodes



Decision Tree classifier for Minimum 50 records at leaf nodes



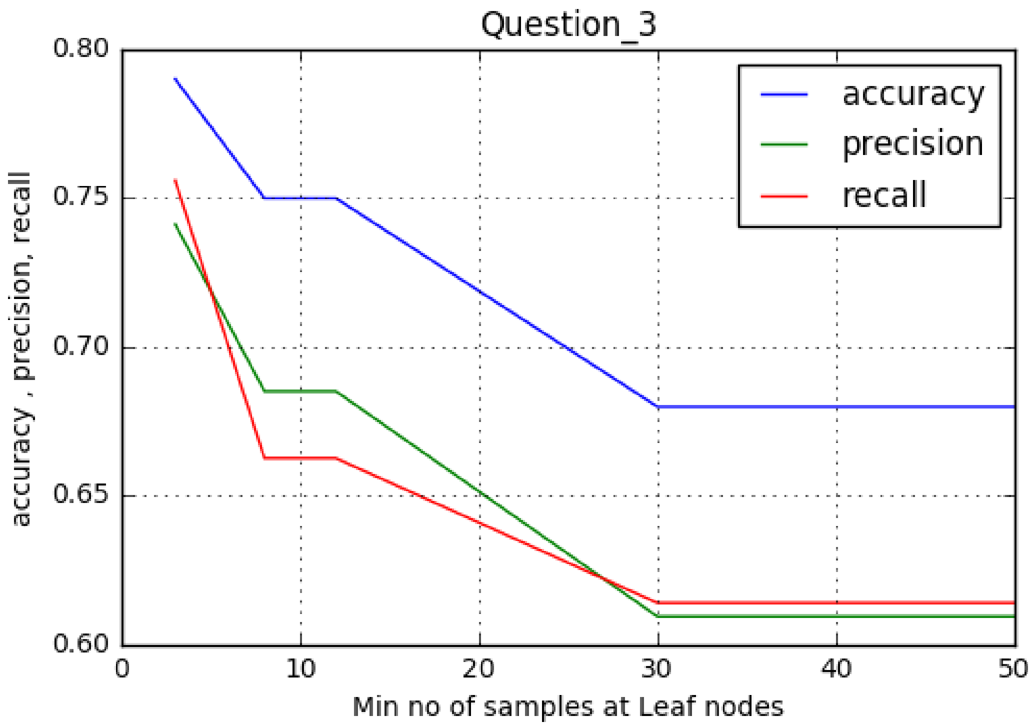
Accuracy, Precision, Recall for Dt with 3 data sets at leaf node are 0.79, 0.741234221598878, 0.7559523809523809

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)

```
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn import tree
import matplotlib.pyplot as plt
import xlsxwriter
import pydotplus
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_recall_fscore_support


def tree_generation_1(f_name):
    accuracy=[]
    precision=[]
    recall=[]
    min_datasets=[3,8,12,30,50]


    # reading the data using read_csv function of the pandas library which directly reads the csv file and creates a datagram of the same
    data = pd.read_csv(f_name)


    # 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
    X = data.values[:, 0:6]      #1st to 6th column
    Y = data.values[:,6]         #last column


    # dividing the records into testing and training data using train_test_split function, which takes percentage as input for test size
    # hence 310 given samples , 210 to be used as test_size so it is 67.77%
    X_train, X_test, y_train, y_test = train_test_split( X, Y, train_size = 0.678, random_state =100)



    #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 3
    dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=3)
    dt_gini.fit(X_train, y_train)


    #predicting the class values by testing the model with testing dataset and storing them
```

```
    y_pred_gini = dt_gini.predict(X_test)

    #calculating the accuracy, precision and recal and storing them
    a = accuracy_score(y_test,y_pred_gini)
    b = precision_score(y_test, y_pred_gini,average='macro')
    c = recall_score(y_test, y_pred_gini,average='macro')
    print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are {},{},{}".format(3,a,b,c))
    accuracy.append(a)
    precision.append(b)
    recall.append(c)


    #exporting the created decision tree to pdf
    dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data)
    nodes = graph.get_node_list()
    colors =  ('blue', 'yellow', 'green', 'red', 'white')
    for node in nodes:
        if node.get_name() not in ('node', 'edge'):
            values = dt_gini.tree_.value[int(node.get_name())][0]
        #color only nodes where only one class is present
        if max(values) == sum(values):
            node.set_fillcolor(colors[np.argmax(values)])
        #mixed nodes get the default color
        else:
            node.set_fillcolor(colors[-1])

    graph.write_png('dt_2c_3min.png')

    #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 8
    dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=8)
    dt_gini.fit(X_train, y_train)


    #predicting the class values by testing the model with testing dataset and storing them
    y_pred_gini = dt_gini.predict(X_test)


    #calculating the accuracy, precision and recal and storing them
    a = accuracy_score(y_test,y_pred_gini)
    b = precision_score(y_test, y_pred_gini,average='macro')
    c = recall_score(y_test, y_pred_gini,average='macro')
    print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are {},{},{}".format(8,a,b,c))
    accuracy.append(a)
    precision.append(b)
    recall.append(c)


    #exporting the created decision tree to pdf
    dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data)
    nodes = graph.get_node_list()
    colors =  ('blue', 'yellow', 'green', 'red', 'white')
    for node in nodes:
        if node.get_name() not in ('node', 'edge'):
            values = dt_gini.tree_.value[int(node.get_name())][0]
            #color only nodes where only one class is present
            if max(values) == sum(values):
                node.set_fillcolor(colors[np.argmax(values)])
            #mixed nodes get the default color
```

```
            else:
                node.set_fillcolor(colors[-1])

        graph.write_png('dt_2c_8min.png')



        #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 12
        dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=12)
        dt_gini.fit(X_train, y_train)



        #predicting the class values by testing the model with testing dataset and storing them
        y_pred_gini = dt_gini.predict(X_test)


        #calculating the accuracy, precision and recal and storing them
        a = accuracy_score(y_test,y_pred_gini)
        b = precision_score(y_test, y_pred_gini,average='macro')
        c = recall_score(y_test, y_pred_gini,average='macro')
        print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are {},{},{}".format(12,a,b,c))
        accuracy.append(a)
        precision.append(b)
        recall.append(c)


         #exporting the created decision tree to pdf
        dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
        graph = pydotplus.graph_from_dot_data(dot_data)
        nodes = graph.get_node_list()
        colors =  ('blue', 'yellow', 'green', 'red', 'white')
        for node in nodes:
            if node.get_name() not in ('node', 'edge'):
                values = dt_gini.tree_.value[int(node.get_name())][0]
            #color only nodes where only one class is present
            if max(values) == sum(values):
                node.set_fillcolor(colors[np.argmax(values)])
            #mixed nodes get the default color
            else:
                node.set_fillcolor(colors[-1])

        graph.write_png('dt_2c_12min.png')


        #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 30
        dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=30)
        dt_gini.fit(X_train, y_train)


        #predicting the class values by testing the model with testing dataset and storing them
        y_pred_gini = dt_gini.predict(X_test)

        #calculating the accuracy, precision and recal and storing them
        a = accuracy_score(y_test,y_pred_gini)
        b = precision_score(y_test, y_pred_gini,average='macro')
        c = recall_score(y_test, y_pred_gini,average='macro')
```

```
        print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are {},{},{}".format(30,a,b,c))
        accuracy.append(a)
        precision.append(b)
        recall.append(c)

        #exporting the created decision tree to pdf
        with open("biomechanical_dt_gini_30.dot", "w") as f:
         f = tree.export_graphviz(dt_gini, out_file=f,feature_names=data.columns.values[0:6])
        !dot -Tpdf "biomechanical_dt_gini_30.dot" -o "biomechanical_dt_gini_30.pdf #exporting the created decision tree to pdf
        dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
        graph = pydotplus.graph_from_dot_data(dot_data)
        nodes = graph.get_node_list()
        colors =  ('blue', 'yellow', 'green', 'red', 'white')
        for node in nodes:
            if node.get_name() not in ('node', 'edge'):
                values = dt_gini.tree_.value[int(node.get_name())][0]
            #color only nodes where only one class is present
            if max(values) == sum(values):
                node.set_fillcolor(colors[np.argmax(values)])
            #mixed nodes get the default color
            else:
                node.set_fillcolor(colors[-1])

        graph.write_png('dt_2c_30min.png')


        #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 50
        dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=50)

        dt_gini.fit(X_train, y_train)



        #predicting the class values by testing the model with testing dataset and storing them
        y_pred_gini = dt_gini.predict(X_test)

        #calculating the accuracy, precision and recal and storing them
        a = accuracy_score(y_test,y_pred_gini)
        b = precision_score(y_test, y_pred_gini,average='macro')
        c = recall_score(y_test, y_pred_gini,average='macro')
        print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are {},{},{}".format(50,a,b,c))
        accuracy.append(a)
        precision.append(b)
        recall.append(c)

        #exporting the created decision tree to pdf
        dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
        graph = pydotplus.graph_from_dot_data(dot_data)
        nodes = graph.get_node_list()
        colors =  ('blue', 'yellow', 'green', 'red', 'white')
        for node in nodes:
            if node.get_name() not in ('node', 'edge'):
                values = dt_gini.tree_.value[int(node.get_name())][0]
            #color only nodes where only one class is present
            if max(values) == sum(values):
                node.set_fillcolor(colors[np.argmax(values)])
            #mixed nodes get the default color
            else:
```

```
                node.set_fillcolor(colors[-1])

        graph.write_png('dt_2c_50min.png')

       #plotting graphs
        fig1 = plt.figure(1)
        plt.plot(min_datasets,accuracy,label="accuracy")
        plt.plot(min_datasets,precision,label="precision")
        plt.plot(min_datasets,recall,label="recall")
        plt.grid(True)
        plt.xlabel('Min no of samples at Leaf nodes')
        plt.ylabel('accuracy , precision, recall')
        plt.legend(['accuracy', 'precision', 'recall'], loc='upper right')
        plt.title('Question_1')
        fig1.savefig('plot_1.png')
        plt.close()
        # add plt.close() after you've saved the figure
        #plt.show()


        fig2 = plt.figure(2)
        plt.plot(precision,recall,'ro')
        plt.xlabel('precision')
        plt.ylabel('recall')
        plt.title('Question_1 precision vs recall')
        fig2.savefig('precision_recall_1.png')
        plt.close()
        print('\n ############################################################################################## \n')

    def tree_generation_2(f_name):
        accuracy=[]
        precision=[]
        recall=[]
        precision_class_normal=[]
        recall_class_normal=[]
        precision_class_spondy=[]
        recall_class_spondy=[]
        precision_class_hernia=[]
        recall_class_hernia=[]
        min_datasets=[3,8,12,30,50]


        # reading the data using read_csv function of the pandas library which directly reads the csv file and creates a datagram of the same
        data = pd.read_csv(f_name)


        # 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
        X = data.values[:, 0:6]      #1st to 6th column
        Y = data.values[:,6]         #last column

        # dividing the records into testing and training data using train_test_split function, which takes percentage as input for test size
        # hence 310 given samples , 210 to be used as test_size so it is 67.77%

        X_train, X_test, y_train, y_test = train_test_split( X, Y, train_size = 0.678, random_state = 60)



        #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 3
        dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=3)
```

```
            dt_gini.fit(X_train, y_train)


            #predicting the class values by testing the model with testing dataset and storing them
            y_pred_gini = dt_gini.predict(X_test)

            #calculating the accuracy, precision and recal and storing them
            a = accuracy_score(y_test,y_pred_gini)
            precision_recall=precision_recall_fscore_support(y_test, y_pred_gini, average=None,labels=['Normal','Spondylolisthesis','Hernia'])
            b=precision_recall[0][0]
            c=precision_recall[0][1]
            d=precision_recall[1][0]
            e=precision_recall[1][1]
            f=precision_recall[2][0]
            g=precision_recall[2][1]
            print("Accuracy for gini Dt with {} data sets at leaf node are {}".format(3,a))
            print("precision , recall for class normal is {}, {}".format(b,c))
            print("precision , recall for class spondy is {}, {}".format(d,e))
            print("precision , recall for class hernia is {}, {} \n".format(f,g))
            accuracy.append(a)
            precision_class_normal.append(b)
            recall_class_normal.append(c)
            precision_class_spondy.append(d)
            recall_class_spondy.append(e)
            precision_class_hernia.append(f)
            recall_class_hernia.append(g)



             #exporting the created decision tree to pdf
            dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
            graph = pydotplus.graph_from_dot_data(dot_data)
            nodes = graph.get_node_list()
            colors =  ('blue', 'yellow', 'green', 'red', 'white')
            for node in nodes:
                if node.get_name() not in ('node', 'edge'):
                    values = dt_gini.tree_.value[int(node.get_name())][0]
                #color only nodes where only one class is present
                if max(values) == sum(values):
                    node.set_fillcolor(colors[np.argmax(values)])
                #mixed nodes get the default color
                else:
                    node.set_fillcolor(colors[-1])

            graph.write_png('dt_3c_3min.png')


            #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 8
            dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=8)

            dt_gini.fit(X_train, y_train)


            #predicting the class values by testing the model with testing dataset and storing them
            y_pred_gini = dt_gini.predict(X_test)

            #calculating the accuracy, precision and recal and storing them
            a = accuracy_score(y_test,y_pred_gini)
            precision_recall=precision_recall_fscore_support(y_test, y_pred_gini, average=None,labels=['Normal','Spondylolisthesis','Hernia'])
```

```
        b=precision_recall[0][0]
        c=precision_recall[0][1]
        d=precision_recall[1][0]
        e=precision_recall[1][1]
        f=precision_recall[2][0]
        g=precision_recall[2][1]
        print("Accuracy for gini Dt with {} data sets at leaf node are {}".format(8,a))
        print("precision , recall for class normal is {}, {}".format(b,c))
        print("precision , recall for class spondy is {}, {}".format(d,e))
        print("precision , recall for class hernia is {}, {} \n".format(f,g))
        accuracy.append(a)
        precision_class_normal.append(b)
        recall_class_normal.append(c)
        precision_class_spondy.append(d)
        recall_class_spondy.append(e)
        precision_class_hernia.append(f)
        recall_class_hernia.append(g)


        #exporting the created decision tree to pdf
        dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
        graph = pydotplus.graph_from_dot_data(dot_data)
        nodes = graph.get_node_list()
        colors =  ('blue', 'yellow', 'green', 'red', 'white')
        for node in nodes:
            if node.get_name() not in ('node', 'edge'):
                values = dt_gini.tree_.value[int(node.get_name())][0]
            #color only nodes where only one class is present
            if max(values) == sum(values):
                node.set_fillcolor(colors[np.argmax(values)])
            #mixed nodes get the default color
            else:
                node.set_fillcolor(colors[-1])

        graph.write_png('dt_3c_8min.png')


        #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 12
        dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=12)
        dt_gini.fit(X_train, y_train)


        #predicting the class values by testing the model with testing dataset and storing them
        y_pred_gini = dt_gini.predict(X_test)

        #calculating the accuracy, precision and recal and storing them
        a = accuracy_score(y_test,y_pred_gini)
        precision_recall=precision_recall_fscore_support(y_test, y_pred_gini, average=None,labels=['Normal','Spondylolisthesis','Hernia'])
        b=precision_recall[0][0]
        c=precision_recall[0][1]
        d=precision_recall[1][0]
        e=precision_recall[1][1]
        f=precision_recall[2][0]
        g=precision_recall[2][1]
        print("Accuracy for gini Dt with {} data sets at leaf node are {}".format(12,a))
        print("precision , recall for class normal is {}, {}".format(b,c))
        print("precision , recall for class spondy is {}, {}".format(d,e))
        print("precision , recall for class hernia is {}, {} \n".format(f,g))
        accuracy.append(a)
```

```
            precision_class_normal.append(b)
            recall_class_normal.append(c)
            precision_class_spondy.append(d)
            recall_class_spondy.append(e)
            precision_class_hernia.append(f)
            recall_class_hernia.append(g)


             #exporting the created decision tree to pdf
            dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
            graph = pydotplus.graph_from_dot_data(dot_data)
            nodes = graph.get_node_list()
            colors =  ('blue', 'yellow', 'green', 'red', 'white')
            for node in nodes:
                if node.get_name() not in ('node', 'edge'):
                    values = dt_gini.tree_.value[int(node.get_name())][0]
                #color only nodes where only one class is present
                if max(values) == sum(values):
                    node.set_fillcolor(colors[np.argmax(values)])
                #mixed nodes get the default color
                else:
                    node.set_fillcolor(colors[-1])

            graph.write_png('dt_3c_12min.png')


            #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 30
            dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=30)

            dt_gini.fit(X_train, y_train)


            #predicting the class values by testing the model with testing dataset and storing them
            y_pred_gini = dt_gini.predict(X_test)

            #calculating the accuracy, precision and recal and storing them
            a = accuracy_score(y_test,y_pred_gini)
            precision_recall=precision_recall_fscore_support(y_test, y_pred_gini, average=None,labels=['Normal','Spondylolisthesis','Hernia'])
            b=precision_recall[0][0]
            c=precision_recall[0][1]
            d=precision_recall[1][0]
            e=precision_recall[1][1]
            f=precision_recall[2][0]
            g=precision_recall[2][1]
            print("Accuracy for gini Dt with {} data sets at leaf node are {}".format(30,a))
            print("precision , recall for class normal is {}, {}".format(b,c))
            print("precision , recall for class spondy is {}, {}".format(d,e))
            print("precision , recall for class hernia is {}, {} \n".format(f,g))
            accuracy.append(a)
            precision_class_normal.append(b)
            recall_class_normal.append(c)
            precision_class_spondy.append(d)
            recall_class_spondy.append(e)
            precision_class_hernia.append(f)
            recall_class_hernia.append(g)

            #exporting the created decision tree to pdf
            dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
            graph = pydotplus.graph_from_dot_data(dot_data)
            nodes = graph.get_node_list()
```

```
            colors =  ('blue', 'yellow', 'green', 'red', 'white')
            for node in nodes:
                if node.get_name() not in ('node', 'edge'):
                    values = dt_gini.tree_.value[int(node.get_name())][0]
                #color only nodes where only one class is present
                if max(values) == sum(values):
                    node.set_fillcolor(colors[np.argmax(values)])
                #mixed nodes get the default color
                else:
                    node.set_fillcolor(colors[-1])

            graph.write_png('dt_3c_30min.png')



            #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 50
            dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=50)
            dt_gini.fit(X_train, y_train)



            #predicting the class values by testing the model with testing dataset and storing them
            y_pred_gini = dt_gini.predict(X_test)

             #calculating the accuracy, precision and recal and storing them
            a = accuracy_score(y_test,y_pred_gini)
            precision_recall=precision_recall_fscore_support(y_test, y_pred_gini, average=None,labels=['Normal','Spondylolisthesis','Hernia'])
            b=precision_recall[0][0]
            c=precision_recall[0][1]
            d=precision_recall[1][0]
            e=precision_recall[1][1]
            f=precision_recall[2][0]
            g=precision_recall[2][1]
            print("Accuracy for gini Dt with {} data sets at leaf node are {}".format(50,a))
            print("precision , recall for class normal is {}, {}".format(b,c))
            print("precision , recall for class Spondy is {}, {}".format(d,e))
            print("precision , recall for class hernia is {}, {} \n".format(f,g))
            accuracy.append(a)
            precision_class_normal.append(b)
            recall_class_normal.append(c)
            precision_class_spondy.append(d)
            recall_class_spondy.append(e)
            precision_class_hernia.append(f)
            recall_class_hernia.append(g)


            #exporting the created decision tree to pdf
            dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
            graph = pydotplus.graph_from_dot_data(dot_data)
            nodes = graph.get_node_list()
            colors =  ('blue', 'yellow', 'green', 'red', 'white')
            for node in nodes:
                if node.get_name() not in ('node', 'edge'):
                    values = dt_gini.tree_.value[int(node.get_name())][0]
                #color only nodes where only one class is present
                if max(values) == sum(values):
                    node.set_fillcolor(colors[np.argmax(values)])
                #mixed nodes get the default color
                else:
                    node.set_fillcolor(colors[-1])
```

```

        graph.write_png('dt_3c_50min.png')

        #plotting graphs
        fig1 = plt.figure(1)
        ax = plt.subplot(111)
        ax.plot(min_datasets,accuracy,label="accuracy")
        ax.plot(min_datasets,precision_class_normal,label="precision_normal")
        ax.plot(min_datasets,recall_class_normal,label="recall_normal")
        ax.plot(min_datasets,precision_class_spondy,label="precision_spondy")
        ax.plot(min_datasets,recall_class_spondy,label="recall_spondy")
        ax.plot(min_datasets,precision_class_hernia,label="precision_hernia")
        ax.plot(min_datasets,recall_class_hernia,label="recall_hernia")
        ax.grid(True)
        ax.set_xlabel('Min no of samples at Leaf nodes')
        #plt.ylabel
        box = ax.get_position()
        ax.set_position([box.x0, box.y0 + box.height * 0.2,box.width, box.height * 0.9])

        ax.legend(['accuracy', 'precision_normal', 'recall_normal','precision_spondy','recall_spondy','precision_hernia','recall_hernia'],loc='upper center', bbox_to_anchor=(0.5, -0.05),
               ncol=3)
        fig1.savefig('plot_2.png')
        plt.close()
        # add plt.close() after you've saved the figure
        #plt.show()


        print('\n ############################################################################################## \n')

    def tree_generation_3(f_name):
        df = pd.read_csv(f_name)
        revised_dataframe = pd.DataFrame()
        bin_boundary=[]
        columns=df.columns
        y=0
        for col in columns[0:6]:
            hist, bin_edges = np.histogram(df[col][1:], bins=4)
            bin_boundary.append(bin_edges)
            col_trans_output=pd.DataFrame(pd.cut(df[col],4,labels=range(4)))
            revised_dataframe = pd.concat([revised_dataframe,col_trans_output], axis=1)


        writer = pd.ExcelWriter('test1.xlsx',engine='xlsxwriter')
        workbook=writer.book
        revised_dataframe.to_excel(writer,sheet_name='Validation1',startrow=0 , startcol=0)
        workbook.close()


        for x in bin_boundary:
            print('Boundaries for {} are {} \n'.format(columns[y],x))
            y=y+1
        print('\n ############################################################################################## \n')
        accuracy=[]
        precision=[]
        recall=[]
        min_datasets=[3,8,12,30,50]

        data=pd.read_csv(f_name)
        #print(revised_dataframe)
```

```
            # 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
            X = revised_dataframe.values[:, 0:6]      #1st to 6th column
            Y = df.values[:,6]         #last column


            # dividing the records into testing and training data using train_test_split function, which takes percentage as input for test size
            # hence 310 given samples , 210 to be used as test_size so it is 67.77%

            X_train, X_test, y_train, y_test = train_test_split( X, Y, train_size = 0.678, random_state = 42)



            #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 3
            dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=3)

            dt_gini.fit(X_train, y_train)


            #predicting the class values by testing the model with testing dataset and storing them
            y_pred_gini = dt_gini.predict(X_test)

            #calculating the accuracy, precision and recal and storing them
            a = accuracy_score(y_test,y_pred_gini)
            b = precision_score(y_test, y_pred_gini,average='macro')
            c = recall_score(y_test, y_pred_gini,average='macro')
            print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are {},{},{}".format(3,a,b,c))
            accuracy.append(a)
            precision.append(b)
            recall.append(c)


             #exporting the created decision tree to pdf
            dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
            graph = pydotplus.graph_from_dot_data(dot_data)
            nodes = graph.get_node_list()
            colors =  ('blue', 'yellow', 'green', 'red', 'white')
            for node in nodes:
                if node.get_name() not in ('node', 'edge'):
                    values = dt_gini.tree_.value[int(node.get_name())][0]
                #color only nodes where only one class is present
                if max(values) == sum(values):
                    node.set_fillcolor(colors[np.argmax(values)])
                #mixed nodes get the default color
                else:
                    node.set_fillcolor(colors[-1])

            graph.write_png('dt_rev_3min.png')


            #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 8
            dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=8)

            dt_gini.fit(X_train, y_train)


            #predicting the class values by testing the model with testing dataset and storing them
            y_pred_gini = dt_gini.predict(X_test)
```

```


            #calculating the accuracy, precision and recal and storing them
            a = accuracy_score(y_test,y_pred_gini)
            b = precision_score(y_test, y_pred_gini,average='macro')
            c = recall_score(y_test, y_pred_gini,average='macro')
            print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are {},{},{}".format(8,a,b,c))
            accuracy.append(a)
            precision.append(b)
            recall.append(c)


            #exporting the created decision tree to pdf
            dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
            graph = pydotplus.graph_from_dot_data(dot_data)
            nodes = graph.get_node_list()
            colors =  ('blue', 'yellow', 'green', 'red', 'white')
            for node in nodes:
                if node.get_name() not in ('node', 'edge'):
                    values = dt_gini.tree_.value[int(node.get_name())][0]
                #color only nodes where only one class is present
                if max(values) == sum(values):
                    node.set_fillcolor(colors[np.argmax(values)])
                #mixed nodes get the default color
                else:
                    node.set_fillcolor(colors[-1])

            graph.write_png('dt_rev_8min.png')



            #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 12
            dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=12)
            dt_gini.fit(X_train, y_train)



            #predicting the class values by testing the model with testing dataset and storing them
            y_pred_gini = dt_gini.predict(X_test)


            #calculating the accuracy, precision and recal and storing them
            a = accuracy_score(y_test,y_pred_gini)
            b = precision_score(y_test, y_pred_gini,average='macro')
            c = recall_score(y_test, y_pred_gini,average='macro')
            print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are {},{},{}".format(12,a,b,c))
            accuracy.append(a)
            precision.append(b)
            recall.append(c)


            #exporting the created decision tree to pdf
            dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
            graph = pydotplus.graph_from_dot_data(dot_data)
            nodes = graph.get_node_list()
            colors =  ('blue', 'yellow', 'green', 'red', 'white')
            for node in nodes:
                if node.get_name() not in ('node', 'edge'):
                    values = dt_gini.tree_.value[int(node.get_name())][0]
```

```
                #color only nodes where only one class is present
                if max(values) == sum(values):
                    node.set_fillcolor(colors[np.argmax(values)])
                #mixed nodes get the default color
                else:
                    node.set_fillcolor(colors[-1])

            graph.write_png('dt_rev_12min.png')


            #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 30
            dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=30)
            dt_gini.fit(X_train, y_train)


            #predicting the class values by testing the model with testing dataset and storing them
            y_pred_gini = dt_gini.predict(X_test)

            #calculating the accuracy, precision and recal and storing them
            a = accuracy_score(y_test,y_pred_gini)
            b = precision_score(y_test, y_pred_gini,average='macro')
            c = recall_score(y_test, y_pred_gini,average='macro')
            print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are {},{},{}".format(30,a,b,c))
            accuracy.append(a)
            precision.append(b)
            recall.append(c)

             #exporting the created decision tree to pdf
            dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
            graph = pydotplus.graph_from_dot_data(dot_data)
            nodes = graph.get_node_list()
            colors =  ('blue', 'yellow', 'green', 'red', 'white')
            for node in nodes:
                if node.get_name() not in ('node', 'edge'):
                    values = dt_gini.tree_.value[int(node.get_name())][0]
                #color only nodes where only one class is present
                if max(values) == sum(values):
                    node.set_fillcolor(colors[np.argmax(values)])
                #mixed nodes get the default color
                else:
                    node.set_fillcolor(colors[-1])

            graph.write_png('dt_rev_30min.png')



            #creating the classifier using DecisionTreeClassifier fucntion with Gini index approach and minimum records per leaf node as 50
            dt_gini = DecisionTreeClassifier(criterion = "gini",min_samples_leaf=50)
            dt_gini.fit(X_train, y_train)



            #predicting the class values by testing the model with testing dataset and storing them
            y_pred_gini = dt_gini.predict(X_test)

            #calculating the accuracy, precision and recal and storing them
            a = accuracy_score(y_test,y_pred_gini)
            b = precision_score(y_test, y_pred_gini,average='macro')
```

```python
            c = recall_score(y_test, y_pred_gini,average='macro')
            print("Accuracy,Precision,Recall for gini Dt with {} data sets at leaf node are {},{},{}".format(50,a,b,c))
            accuracy.append(a)
            precision.append(b)
            recall.append(c)


             #exporting the created decision tree to pdf
            dot_data = tree.export_graphviz(dt_gini,feature_names=data.columns.values[0:6],out_file=None,filled=True,rounded=True,special_characters=True)
            graph = pydotplus.graph_from_dot_data(dot_data)
            nodes = graph.get_node_list()
            colors =  ('blue', 'yellow', 'green', 'red', 'white')
            for node in nodes:
                if node.get_name() not in ('node', 'edge'):
                    values = dt_gini.tree_.value[int(node.get_name())][0]
                #color only nodes where only one class is present
                if max(values) == sum(values):
                    node.set_fillcolor(colors[np.argmax(values)])
                #mixed nodes get the default color
                else:
                    node.set_fillcolor(colors[-1])

            graph.write_png('dt_rev_50min.png')

            #plotting graphs
            fig1 = plt.figure(1)
            plt.plot(min_datasets,accuracy,label="accuracy")
            plt.plot(min_datasets,precision,label="precision")
            plt.plot(min_datasets,recall,label="recall")
            plt.grid(True)
            plt.xlabel('Min no of samples at Leaf nodes')
            plt.ylabel('accuracy , precision, recall')
            plt.legend(['accuracy', 'precision', 'recall'], loc='upper right')
            plt.title('Question_3')
            fig1.savefig('plot_3.png')
            plt.close()
            # add plt.close() after you've saved the figure
            #plt.show()


            fig2 = plt.figure(2)
            plt.plot(precision,recall,'ro')
            plt.xlabel('precision')
            plt.ylabel('recall')
            plt.title('Question_3 precision vs recall')
            fig2.savefig('precision_recall_3.png')
            plt.close()
        tree_generation_1("Biomechanical_Data_column_2C_weka.csv")
        tree_generation_2("BiomechanicalData_column_3C_weka.csv")
        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 are 0.79,0.7738095238095237,0.7527472527472527
Accuracy,Precision,Recall for gini Dt with 30 data sets at leaf node are 0.73,0.7083333333333333,0.7197802197802198
```

```
Accuracy,Precision,Recall for gini Dt with 50 data sets at leaf node ar
e 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, 0.9215686274
509803
precision , recall for class hernia is 0.7272727272727273, 0.9306930693
069307

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, 0.9803921568
627451
precision , recall for class hernia is 0.7586206896551724, 0.9900990099
0099

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, 0.9803921568
627451
precision , recall for class hernia is 0.6538461538461539, 0.9900990099
0099

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, 0.9803921568
627451
precision , recall for class hernia is 0.6451612903225806, 0.9900990099
0099

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, 0.9803921568
627451
precision , recall for class hernia is 0.6909090909090909, 0.9900990099
0099


 #######################################################################
#######################

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
```