ASSIGNMENT NO.1

RACHIT AGGARWAL - M12506500

Importing required python libraries

In [2]:
```
%matplotlib inline
import matplotlib.pyplot as plt  # to plot graphs
import pandas as pd              # to make dataframes(tables)
import numpy as np               # for numerical calculations
from sklearn.utils import shuffle   # to shuffle data
from sklearn.tree import DecisionTreeClassifier  # to import decision tree fun
ction
from sklearn.metrics import accuracy_score       # to import accuracy function
from sklearn import tree                         # to built trees
from sklearn.metrics import precision_recall_fscore_support  # to import funct
ion for Precision, Recall and Fscore
from IPython.display import Image                            # to print images
from sklearn.tree import export_graphviz                     # to vizualize gra
phs
import pydotplus
import collections
```

Answer 1

Importing Data2 in pandas dataframe & having look at the data

In [95]:
```
data2 = pd.read_csv('C:/Users/rachi/Documents/UC Study Material/Sem 2/Intellig
ent Data Analysis/Assignments/Assignment 1/data2.csv')
data2.head()
```

Out[95]:

	pelvic_incidence	pelvic_tilt numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degr
0	63.027818	22.552586	39.609117	40.475232	98.672917	-0.25
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.56
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.53
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.2
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.91

In [96]:
```
list(data2.keys()) # column names
```

Out[96]:
```
['pelvic_incidence',
 'pelvic_tilt numeric',
 'lumbar_lordosis_angle',
 'sacral_slope',
 'pelvic_radius',
 'degree_spondylolisthesis',
 'class']
```

Shuffling the data file

In [97]:
```
data2 = shuffle(data2)
data2.head()
```

Out[97]:

	pelvic_incidence	pelvic_tilt numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	de
298	66.507179	20.897672	31.727471	45.609507	128.902905	1.
70	72.560702	17.385191	52.000000	55.175511	119.193724	32
27	43.580964	16.508884	47.000000	27.072080	109.271634	8.
154	41.187770	5.792974	42.867392	35.394796	103.348880	27
48	40.557357	17.977784	34.000000	22.579573	121.046246	-1

Dividing training and testing data

In [98]:
```
X = data2.loc[:,'pelvic_incidence':'degree_spondylolisthesis'] # features
y = data2.loc[:,'class'] # target label
X.head()
```

Out[98]:

	pelvic_incidence	pelvic_tilt numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	de
298	66.507179	20.897672	31.727471	45.609507	128.902905	1.
70	72.560702	17.385191	52.000000	55.175511	119.193724	32
27	43.580964	16.508884	47.000000	27.072080	109.271634	8.
154	41.187770	5.792974	42.867392	35.394796	103.348880	27
48	40.557357	17.977784	34.000000	22.579573	121.046246	-1

In [99]:
```
y.head()
```

Out[99]:
```
298      Normal
70     Abnormal
27     Abnormal
154    Abnormal
48     Abnormal
Name: class, dtype: object
```

In [100]:
```
X_train, X_test  = X[:210], X[210:] # training and testing features
```

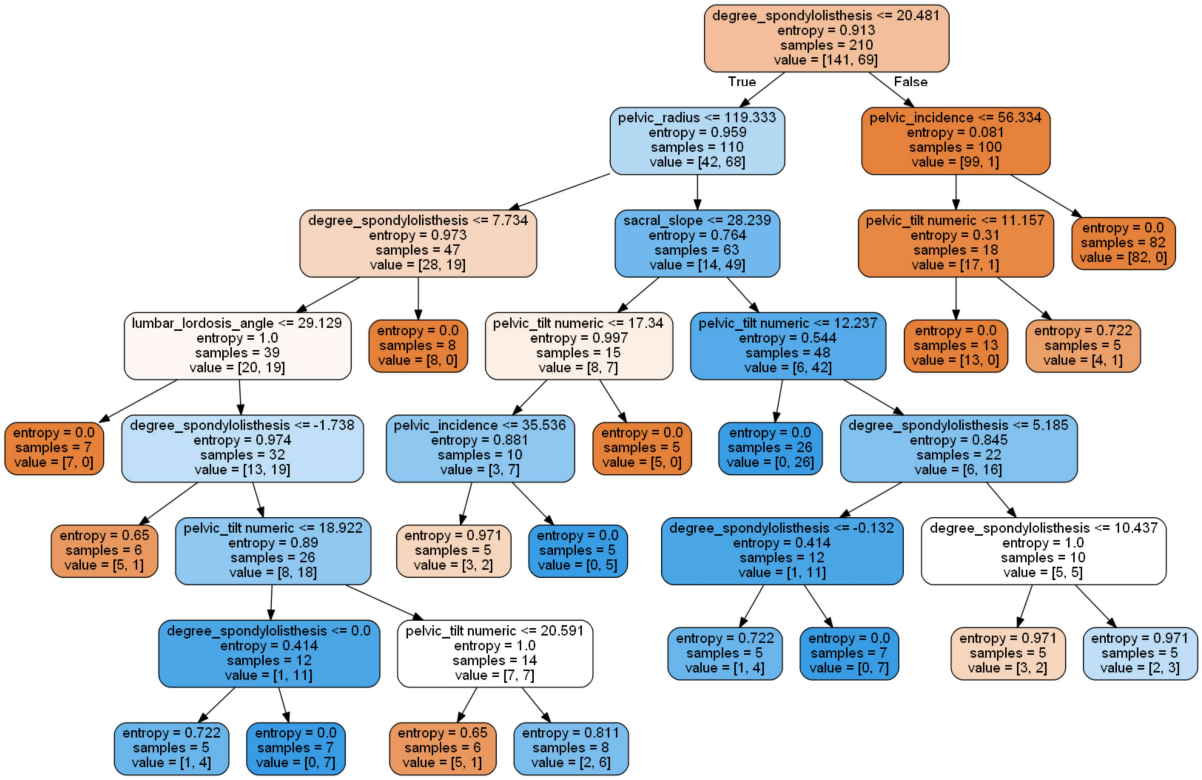
In [101]:
```
y_train, y_test = y[:210], y[210:] # training and testing labels
```

1 a). Applying Decision Tree algorithm

Decision Tree with min_leaf_sample = 5

In [102]:
```
feats = ['pelvic_incidence','pelvic_tilt numeric','lumbar_lordosis_angle','sac
ral_slope','pelvic_radius',
         'degree_spondylolisthesis']
decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state = 1
00, min_samples_leaf = 5)
fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation gain
dot_data = tree.export_graphviz(fit, feature_names = feats, out_file=None, fil
led=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

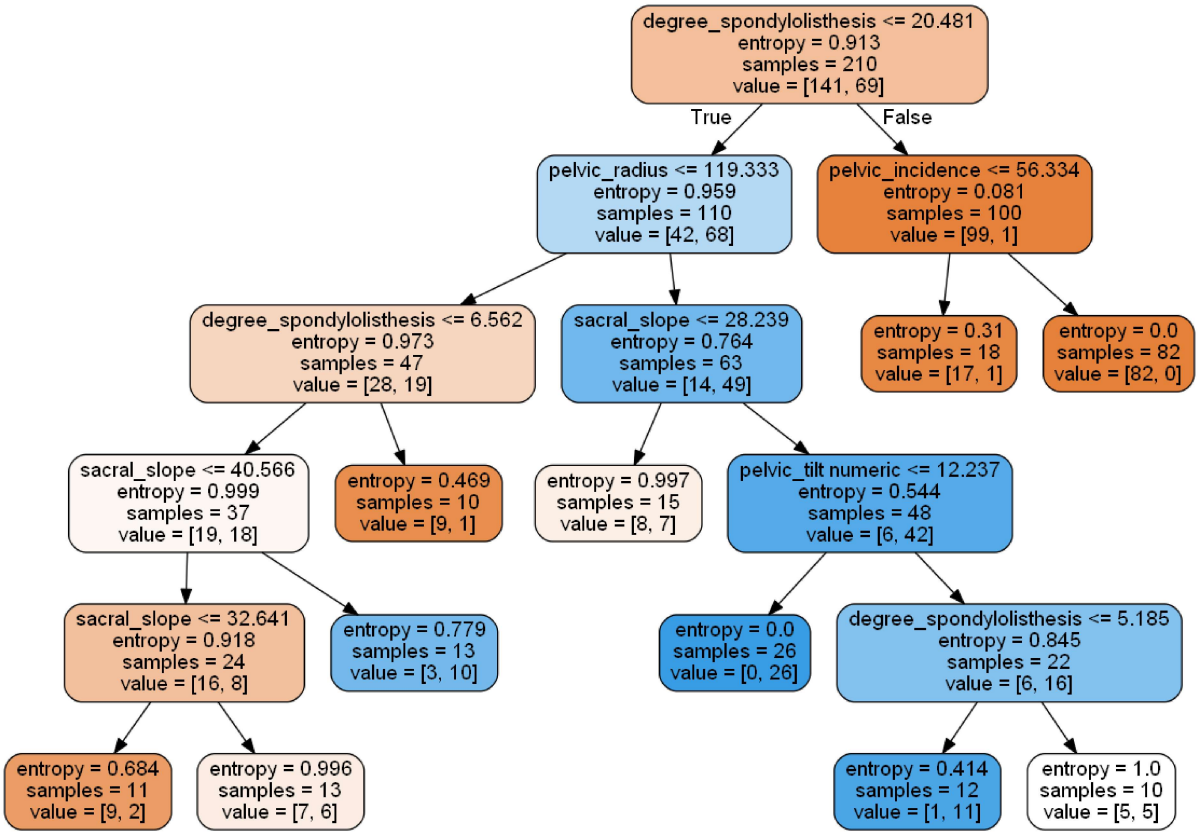
Out[102]:



Decision Tree with min_leaf_sample = 10

In [103]:
```
decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state = 1
00, min_samples_leaf = 10)
fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation gain
dot_data = tree.export_graphviz(fit, feature_names = feats, out_file=None, fil
led=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

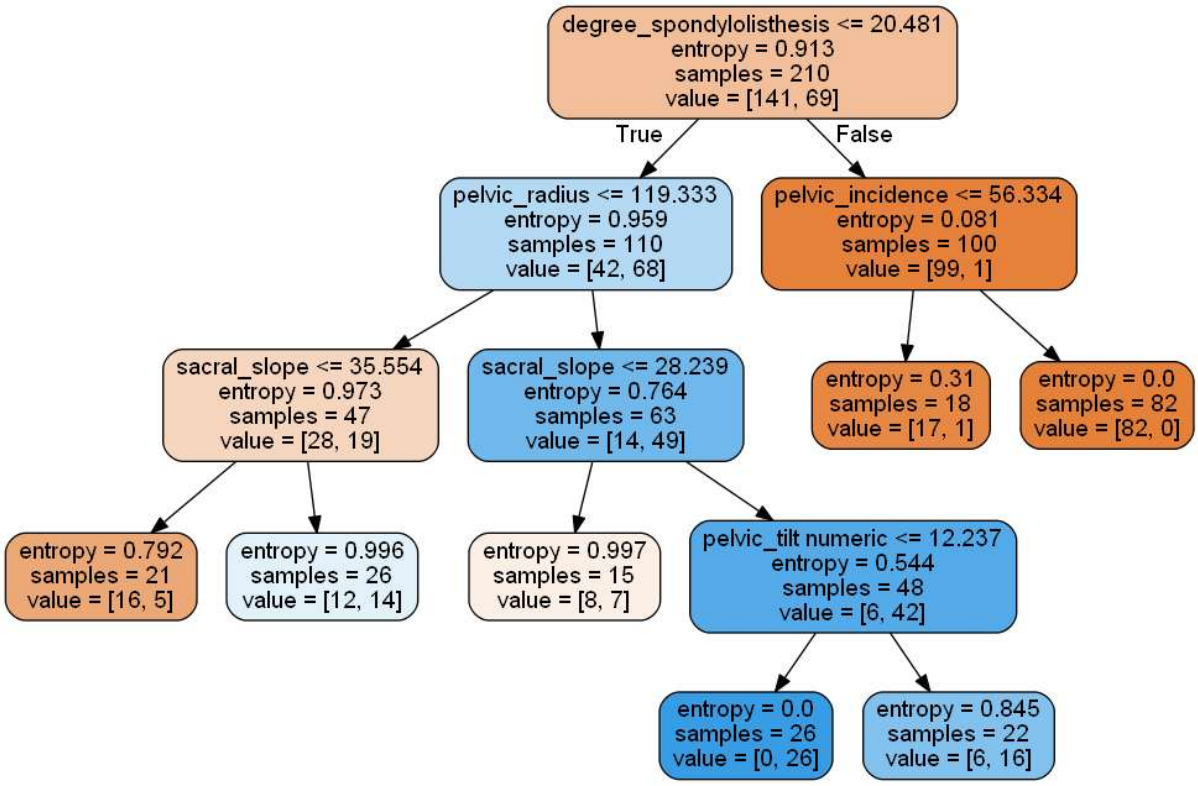
Out[103]:



Decision Tree with min_leaf_sample = 15

In [104]:
```
decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state = 1
00, min_samples_leaf = 15)
fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation gain
dot_data = tree.export_graphviz(fit, feature_names = feats, out_file=None, fil
led=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

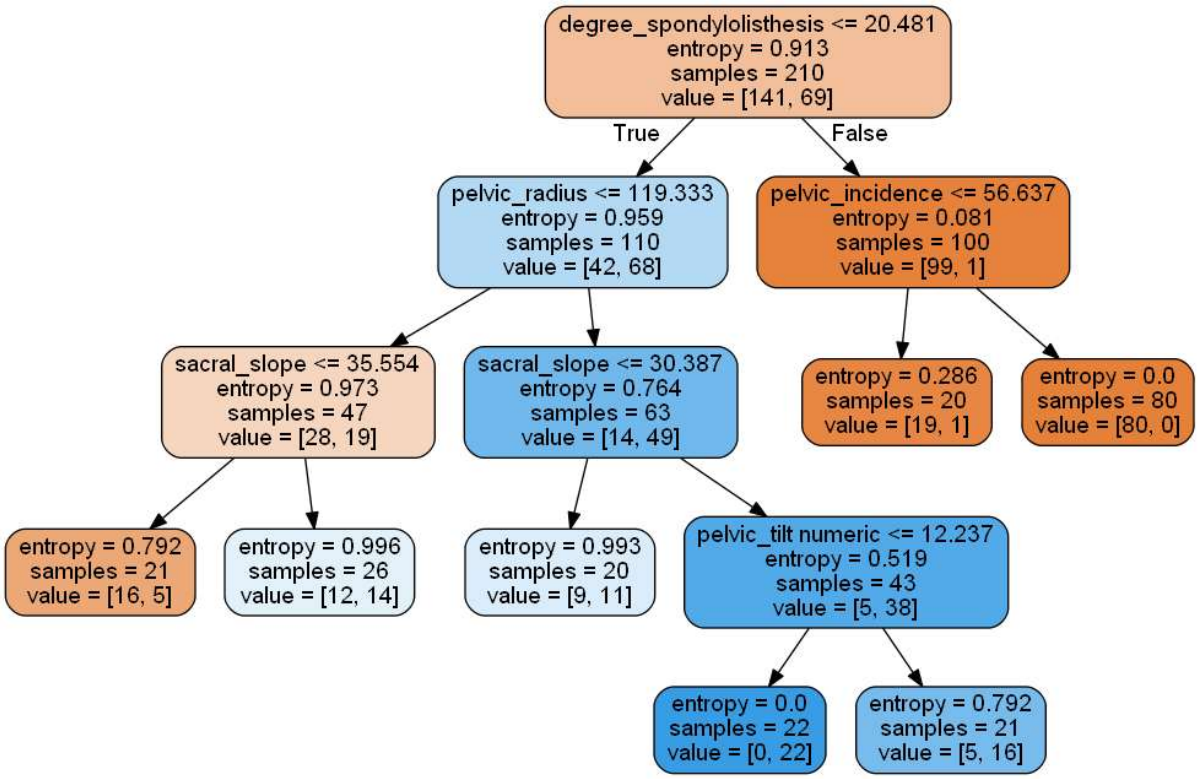
Out[104]:



Decision Tree with min_leaf_sample = 20

In [105]:
```
decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state = 1
00, min_samples_leaf = 20)
fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation gain
dot_data = tree.export_graphviz(fit, feature_names = feats, out_file=None, fil
led=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

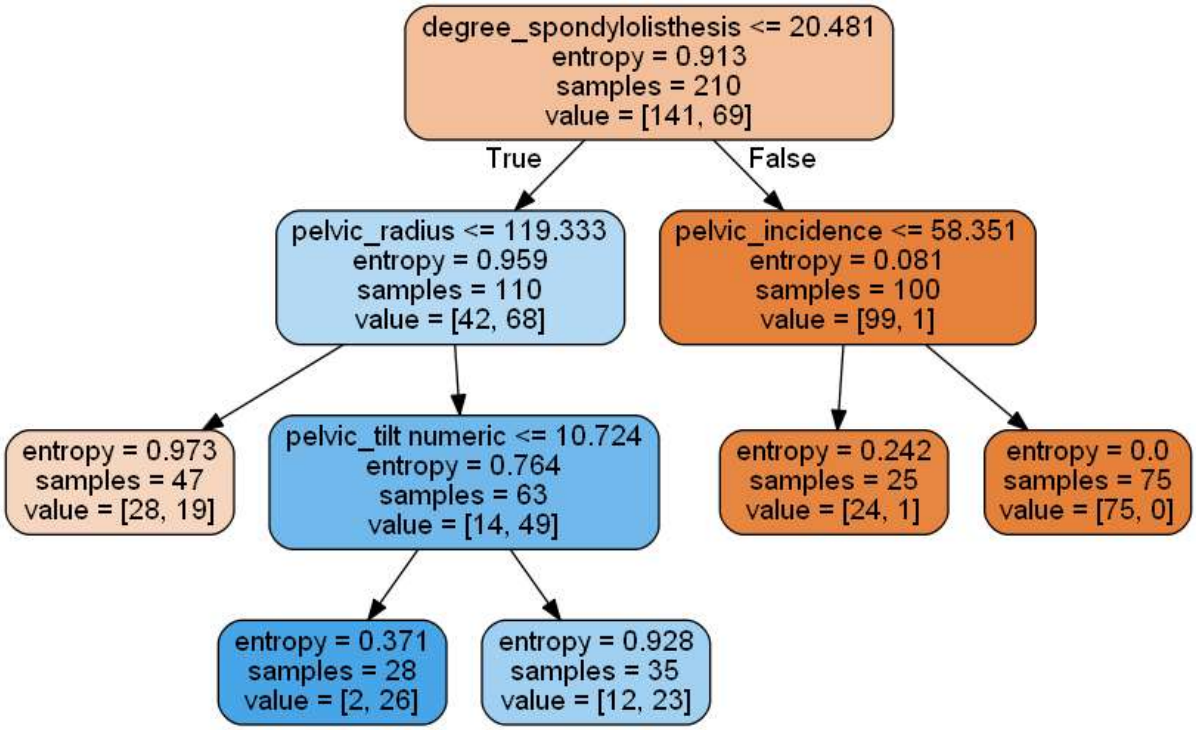
Out[105]:



Decision Tree with min_leaf_sample = 25

In [106]:
```
decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state = 1
00, min_samples_leaf = 25)
fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation gain
dot_data = tree.export_graphviz(fit, feature_names = feats, out_file=None, fil
led=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Out[106]:



As we increase the number of minimum sample for leaf node the depth of the decision tree decreases. Decision tree with 10 minimum sample at leaf has more depth while the tree with 25 minimum sample nodes at leaf is the shallowest

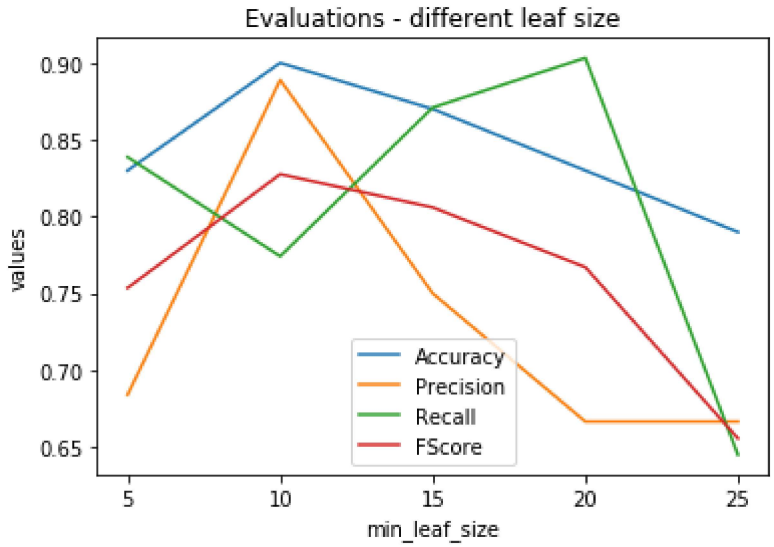
1 b). Evaluation of Decision Tree

In [109]:
```
sample_leaf = [5, 10, 15, 20, 25]  # List with min sample leaf values
Precision = []  # List to store precision
Recall = []  # List to store Recall
FScore = []  # List to store Fscore
Accuracy = []  # List to store Accuracy
for x in sample_leaf:
    decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state
 = 100, min_samples_leaf = x)
    fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation
 gain
    pred = fit.predict(X_test)            # Predicting for text features
    acc = accuracy_score(y_test, pred, normalize=True, sample_weight=None)  #
 Accuracy
    evaluate = precision_recall_fscore_support(y_test, pred, pos_label='Norma
l', average='binary')  #Precision, Recall & FScore
    print("\nMinimum Sample Leaf: " + str(x) + "\nAccuracy: " + str(acc) + "\n
Precision: " + str(evaluate[0]) + "\nRecall: "
          + str(evaluate[1]) + "\nFscore: " + str(evaluate[2]) + "\n")

    Precision.append(evaluate[0])
    Recall.append(evaluate[1])
    FScore.append(evaluate[2])
    Accuracy.append(acc)
```

```
Minimum Sample Leaf: 5
Accuracy: 0.83
Precision: 0.684210526316
Recall: 0.838709677419
Fscore: 0.753623188406


Minimum Sample Leaf: 10
Accuracy: 0.9
Precision: 0.888888888889
Recall: 0.774193548387
Fscore: 0.827586206897


Minimum Sample Leaf: 15
Accuracy: 0.87
Precision: 0.75
Recall: 0.870967741935
Fscore: 0.805970149254


Minimum Sample Leaf: 20
Accuracy: 0.83
Precision: 0.666666666667
Recall: 0.903225806452
Fscore: 0.767123287671


Minimum Sample Leaf: 25
Accuracy: 0.79
Precision: 0.666666666667
Recall: 0.645161290323
Fscore: 0.655737704918
```

Plotting Evaluation

In [110]:
```
plt.plot(sample_leaf, Accuracy,label='Accuracy')
plt.plot(sample_leaf, Precision,label='Precision')
plt.plot(sample_leaf, Recall,label='Recall')
plt.plot(sample_leaf ,FScore,label='FScore')
plt.legend()
plt.xticks(sample_leaf,['5','10','15','20','25'])
plt.title("Evaluations - different leaf size")
plt.xlabel("min_leaf_size")
plt.ylabel("values")

plt.show()
```



As we increase the number of minimum sample at leaf node the Precision, Recall, Fscore and Accuracy all declines. The measures reaches to the highest at leaf size 10, but on increasing the values further the measurements decrease and will underfit the data

1 c). Limiting min_sample_leaf to 10

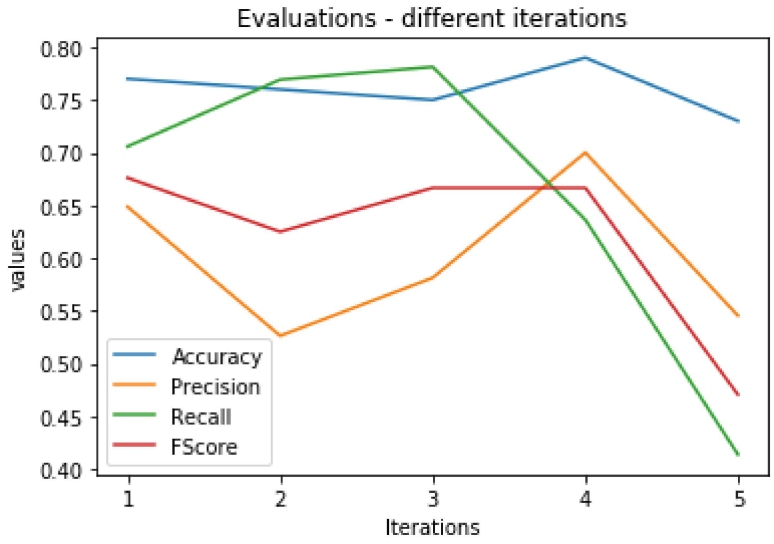
Execution Decision Tree for multiple iterations

In [40]:
```
Precision = []
Recall = []
FScore = []
Accuracy = []
for a in range(5):
    data2 = shuffle(data2)
    X = data2.loc[:,'pelvic_incidence':'degree_spondylolisthesis']
    y = data2.loc[:,'class']
    X_train,X_test,y_train,y_test = X[:210] , X[210:], y[:210] , y[210:]
    decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state
 = 100, min_samples_leaf=10)
    fit = decision_tree.fit(X_train,y_train)
    pred = fit.predict(X_test)
    acc = accuracy_score(y_test, pred, normalize=True, sample_weight=None)
    evaluate = precision_recall_fscore_support(y_test, pred, pos_label='Norma
l', average='binary')
    print("\nIteration: " + str(a+1) + "\nAccuracy: " + str(acc) + "\nPrecisio
n: " + str(evaluate[0]) + "\nRecall: "
          + str(evaluate[1]) + "\nFscore: " + str(evaluate[2]) + "\n")

    Precision.append(evaluate[0])
    Recall.append(evaluate[1])
    FScore.append(evaluate[2])
    Accuracy.append(acc)
```

```
Iteration: 1
Accuracy: 0.82
Precision: 0.705882352941
Recall: 0.75
Fscore: 0.727272727273


Iteration: 2
Accuracy: 0.82
Precision: 0.75
Recall: 0.705882352941
Fscore: 0.727272727273


Iteration: 3
Accuracy: 0.85
Precision: 0.833333333333
Recall: 0.714285714286
Fscore: 0.769230769231


Iteration: 4
Accuracy: 0.79
Precision: 0.769230769231
Recall: 0.571428571429
Fscore: 0.655737704918


Iteration: 5
Accuracy: 0.79
Precision: 0.611111111111
Recall: 0.758620689655
Fscore: 0.676923076923
```

Plotting evaluations for all iterations

In [52]:
```
iterations = [1,2,3,4,5]
plt.plot(iterations, Accuracy,label='Accuracy')
plt.plot(iterations, Precision,label='Precision')
plt.plot(iterations, Recall,label='Recall')
plt.plot(iterations ,FScore,label='FScore')
plt.legend()
plt.xticks(iterations,['1','2','3','4','5'])
plt.title("Evaluations - different iterations")
plt.xlabel("Iterations")
plt.ylabel("values")

plt.show()
```

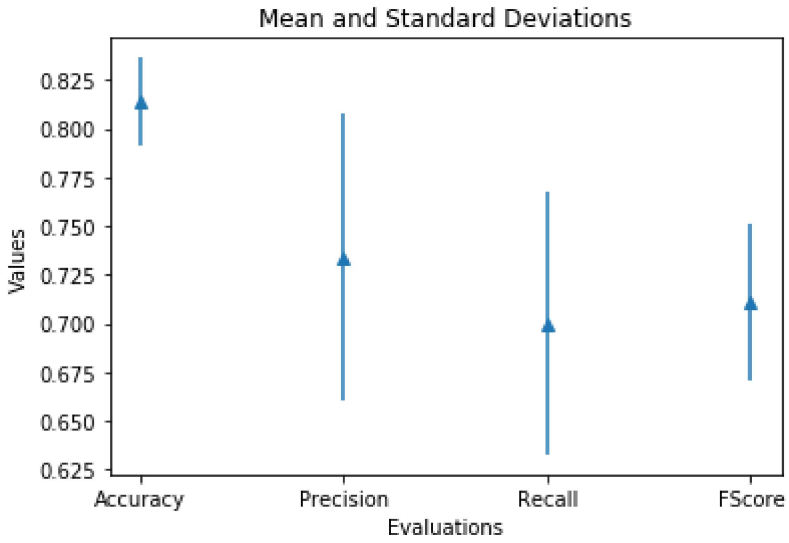


Calculating Aaverage and standard deviation of Precision, Recall, FScore and Accuracy

In [98]:
```
# average Accuracy, Precision, Recall and FScore
average = []
avg_accuracy = np.mean(Accuracy)
avg_precision = np.mean(Precision)
avg_recall = np.mean(Recall)
avg_fscore = np.mean(FScore)
average.append(avg_accuracy)
average.append(avg_precision)
average.append(avg_recall)
average.append(avg_fscore)

# standard deviation for Accuracy, Precision and Recall
stdev = []
std_accuracy = np.std(Accuracy)
std_precision = np.std(Precision)
std_recall = np.std(Recall)
std_fscore = np.std(FScore)
stdev.append(std_accuracy)
stdev.append(std_precision)
stdev.append(std_recall)
stdev.append(std_fscore)
```

Plotting average and standard deviations for Precision, Recall, Fscore and Accuracy

In [115]:
```
xax = ["Accuracy","Precision", "Recall", "FScore"]
step = [1,2,3,4]
plt.figsize = (30,15)
plt.errorbar(step, average, stdev, linestyle='None', marker='^')
plt.xlabel("Evaluations")
plt.ylabel("Values")
plt.title("Mean and Standard Deviations")
plt.xticks(step, xax)
plt.show()
```



The predictions of decision tree algorithm totaly depends on the input data. Shuffling the input training data in each iteration produces completely different trees and hence variations in the Precision, Recall, Fscore and Accuracy can be noticed. According to the graphs above there is not much variation in accuracy but a lot of variations in Precision, Recall and Fscore.

Answer 2

Importing data3 into pandas dataframe

In [75]:
```
data3 = pd.read_csv('C:/Users/rachi/Documents/UC Study Material/Sem 2/Intellig
ent Data Analysis/Assignments/Assignment 1/data3.csv')
data3['class'].value_counts()
```

Out[75]:
```
Spondylolisthesis    150
Normal               100
Hernia                60
Name: class, dtype: int64
```

Shuffling the data

In [76]:
```
data3 = shuffle(data3)
X = data3.loc[:,'pelvic_incidence':'degree_spondylolisthesis']
y = data3.loc[:,'class']
```

Seperating testing and training data

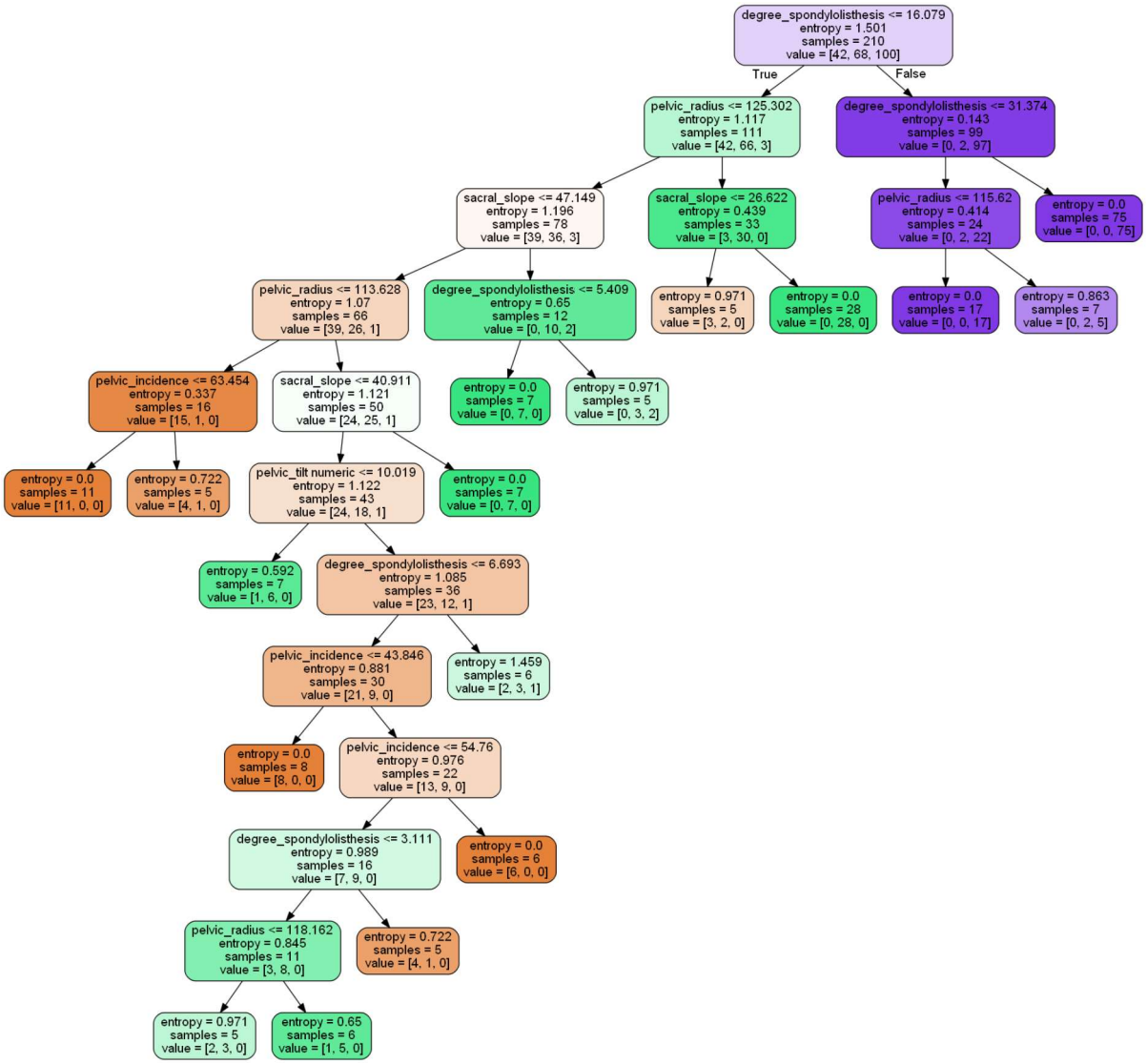
In [77]:
```
X_train,X_test = X[:210] , X[210:]
y_train,y_test = y[:210] , y[210:]
```

2 a). Plotting Decision Trees

Decision Tree with min_leaf_sample = 5

In [78]:
```
feats = ['pelvic_incidence','pelvic_tilt numeric','lumbar_lordosis_angle',
'sacral_slope','pelvic_radius','degree_spondylolisthesis']
decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state
= 100, min_samples_leaf = 5)
fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation g
ain
dot_data = tree.export_graphviz(fit, feature_names = feats, out_file=None,
filled=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

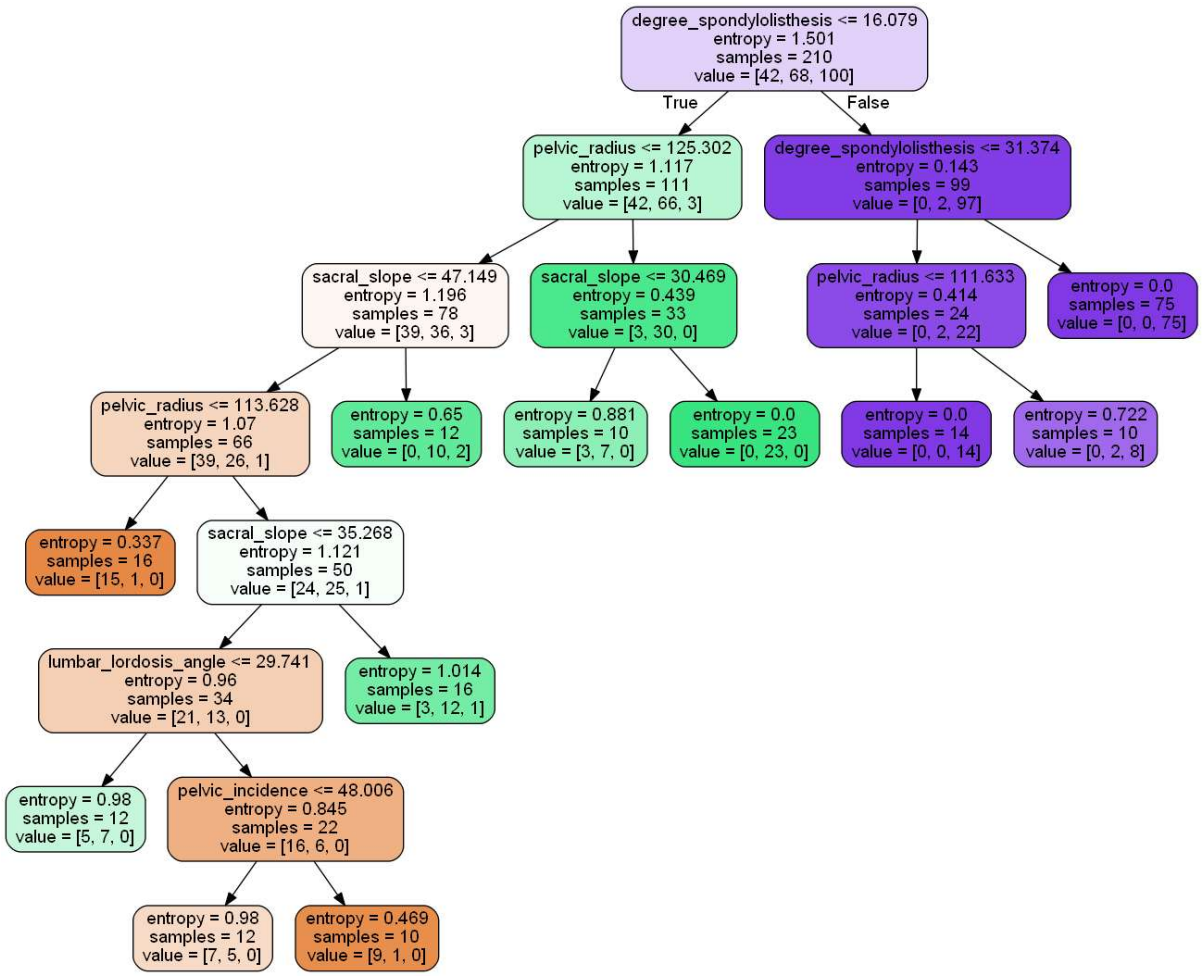
Out[78]:



Decision Tree with min_leaf_sample = 10

In [79]:
```
feats = ['pelvic_incidence','pelvic_tilt numeric','lumbar_lordosis_angle',
'sacral_slope','pelvic_radius','degree_spondylolisthesis']
decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state
= 100, min_samples_leaf = 10)
fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation g
ain
dot_data = tree.export_graphviz(fit, feature_names = feats, out_file=None,
filled=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

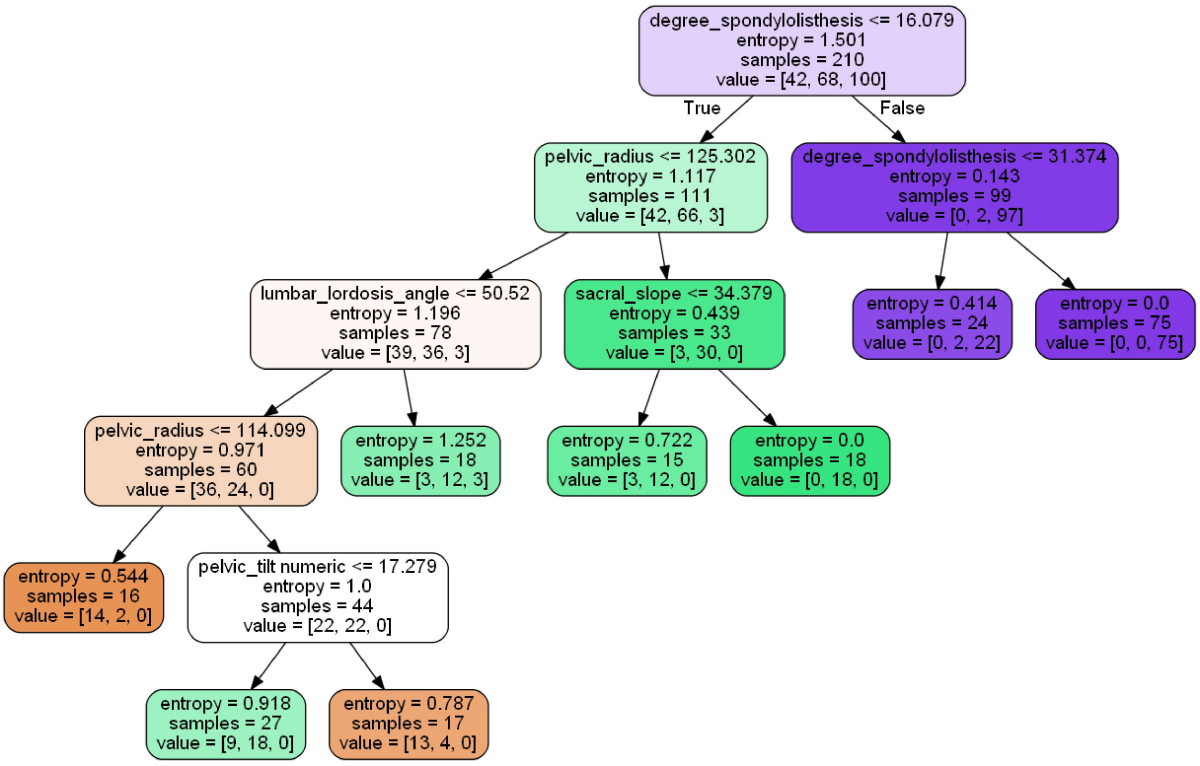
Out[79]:



Decision Tree with min_leaf_sample = 15

In [80]:
```
feats = ['pelvic_incidence','pelvic_tilt numeric','lumbar_lordosis_angle','sac
ral_slope','pelvic_radius','degree_spondylolisthesis']
decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state = 1
00, min_samples_leaf = 15)
fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation gain
dot_data = tree.export_graphviz(fit, feature_names = feats, out_file=None, fil
led=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

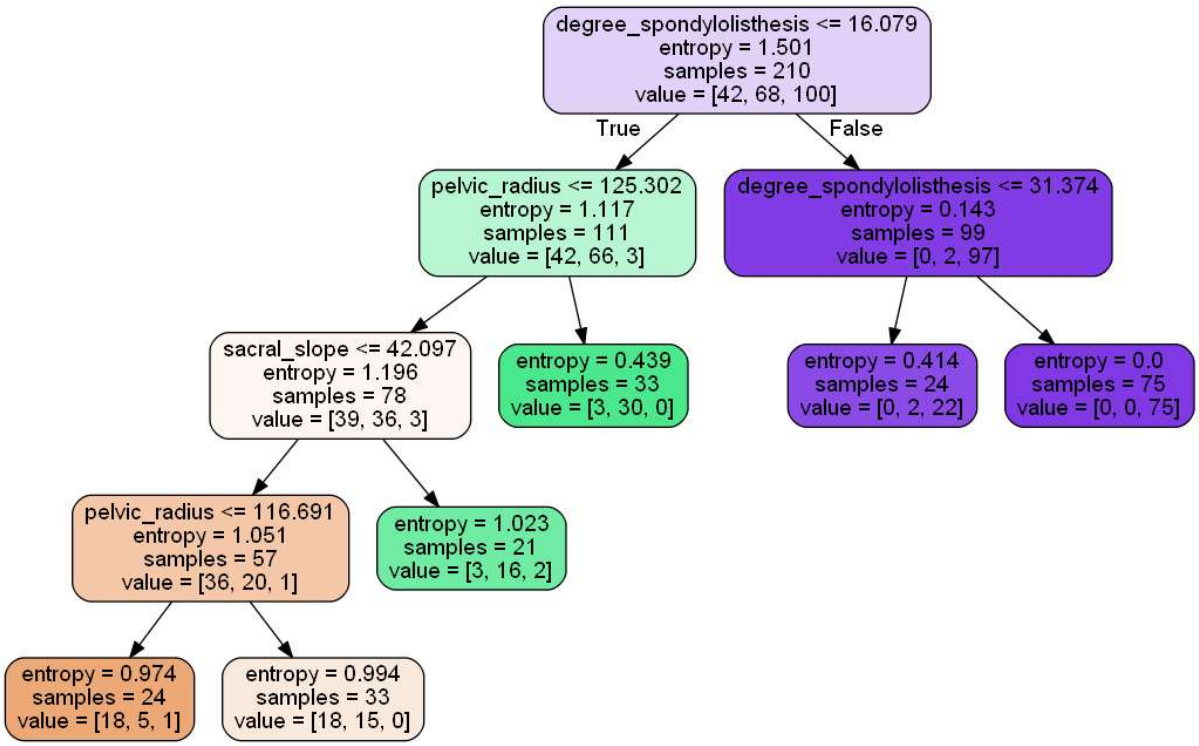
Out[80]:



Decision Tree with min_leaf_sample = 20

In [81]:
```
feats = ['pelvic_incidence','pelvic_tilt numeric','lumbar_lordosis_angle','sac
ral_slope','pelvic_radius','degree_spondylolisthesis']
decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state = 1
00, min_samples_leaf = 20)
fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation gain
dot_data = tree.export_graphviz(fit, feature_names = feats, out_file=None, fil
led=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

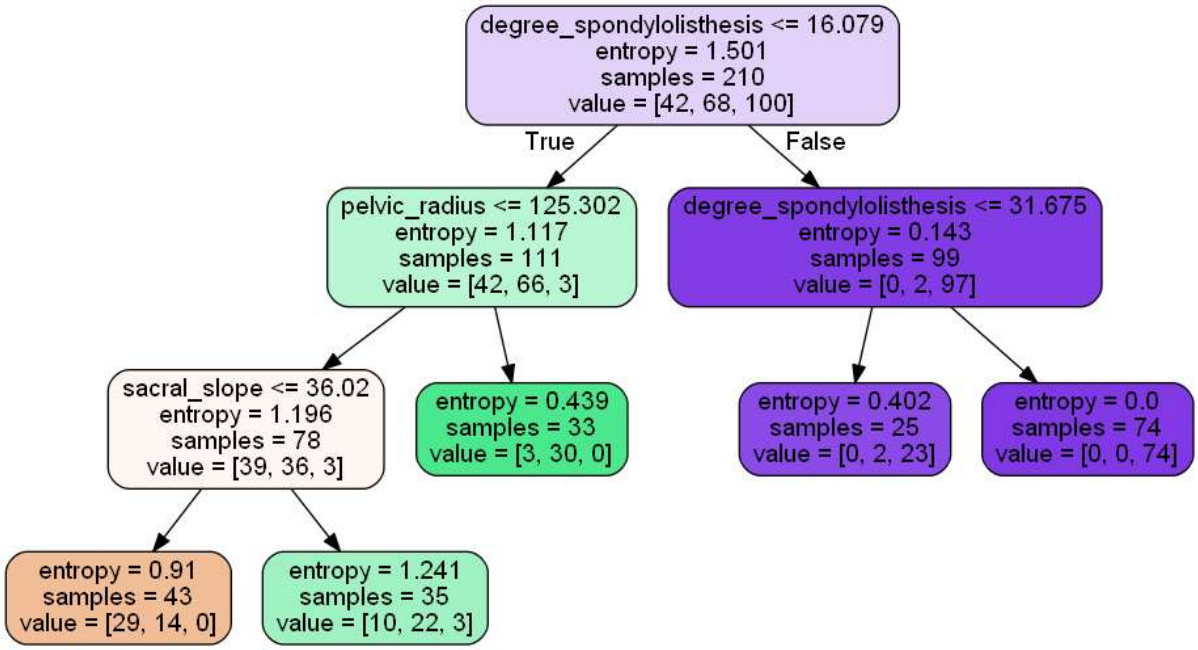
Out[81]:



Decision Tree with min_leaf_sample = 25

In [82]:
```
feats = ['pelvic_incidence','pelvic_tilt numeric','lumbar_lordosis_angle','sac
ral_slope','pelvic_radius','degree_spondylolisthesis']
decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state = 1
00, min_samples_leaf = 25)
fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation gain
dot_data = tree.export_graphviz(fit, feature_names = feats, out_file=None, fil
led=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Out[82]:



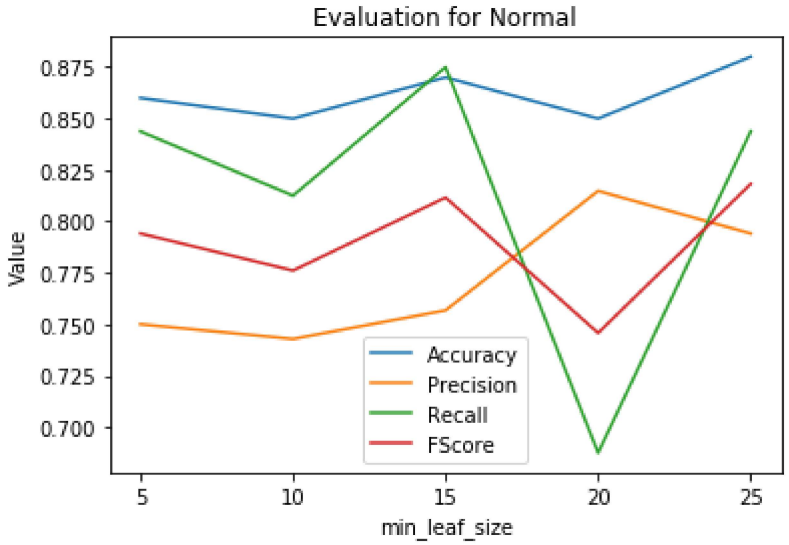
2 b). Evaluations at different min_leaf_sample for all 3 classes

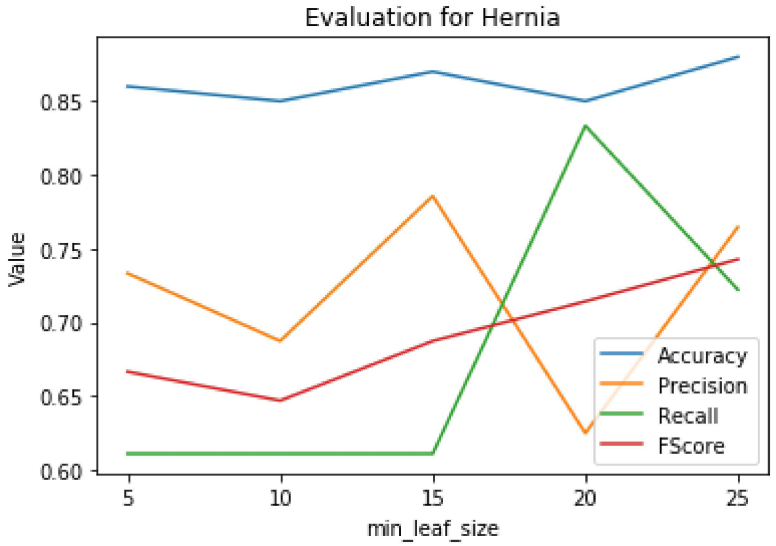
In [83]:
```
sample_leaf = [5,10,15,20,25]
Precision = []
Recall = []
FScore = []
Accuracy = []
for i in a:
    decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state
 = 100, min_samples_leaf=i)
    fit = decision_tree.fit(X_train,y_train)
    pred = fit.predict(X_test)
    acc = accuracy_score(y_test, pred, normalize=True, sample_weight=None)
    result = precision_recall_fscore_support(y_test, pred, average=None, label
s=['Normal', 'Hernia', 'Spondylolisthesis'])
    print("Min_sample_leaf:" + str(i) + "\n\t Accuracy:\t" +str(acc) + "\n\t P
recision: " +str(result[0])
                          +"\n\t Recall: "  +str(result[1]) + "\n\t F1 Score:
 " + str(result[2]))
    Precision.append(result[0])
    Recall.append(result[1])
    FScore.append(result[2])
    Accuracy.append(acc)
```

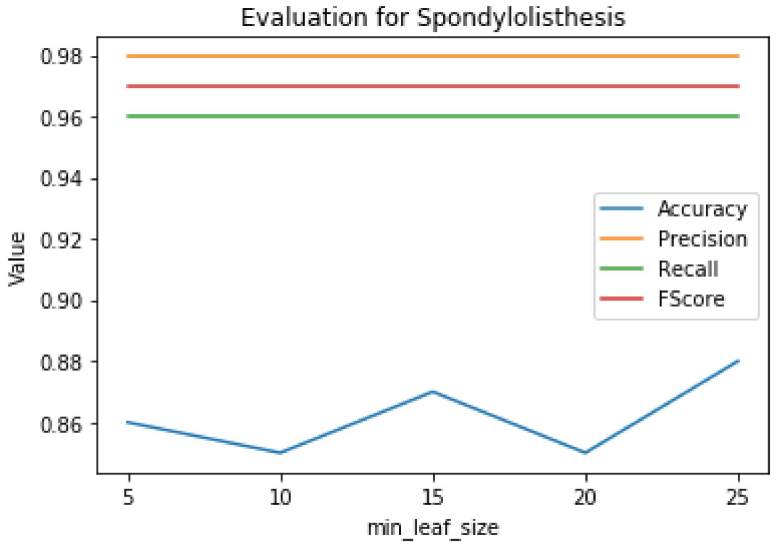
```
Min_sample_leaf:5
         Accuracy:      0.86
         Precision: [ 0.75        0.73333333  0.97959184]
         Recall: [ 0.84375     0.61111111  0.96      ]
         F1 Score: [ 0.79411765  0.66666667  0.96969697]
Min_sample_leaf:10
         Accuracy:      0.85
         Precision: [ 0.74285714  0.6875      0.97959184]
         Recall: [ 0.8125      0.61111111  0.96      ]
         F1 Score: [ 0.7761194   0.64705882  0.96969697]
Min_sample_leaf:15
         Accuracy:      0.87
         Precision: [ 0.75675676  0.78571429  0.97959184]
         Recall: [ 0.875       0.61111111  0.96      ]
         F1 Score: [ 0.8115942   0.6875      0.96969697]
Min_sample_leaf:20
         Accuracy:      0.85
         Precision: [ 0.81481481  0.625       0.97959184]
         Recall: [ 0.6875      0.83333333  0.96      ]
         F1 Score: [ 0.74576271  0.71428571  0.96969697]
Min_sample_leaf:25
         Accuracy:      0.88
         Precision: [ 0.79411765  0.76470588  0.97959184]
         Recall: [ 0.84375     0.72222222  0.96      ]
         F1 Score: [ 0.81818182  0.74285714  0.96969697]
```

Plotting Evaluations

In [84]:
```
Precision_df = pd.DataFrame(Precision, columns=('Normal','Hernia','Spondylolis
thesis'))
Recall_df = pd.DataFrame(Recall, columns=('Normal','Hernia','Spondylolisthesi
s'))
FScore_df = pd.DataFrame(FScore, columns=('Normal','Hernia','Spondylolisthesi
s'))
classname = ['Normal','Hernia','Spondylolisthesis']

for i in range(3):
    plt.plot(sample_leaf,Accuracy,label='Accuracy')
    plt.plot(sample_leaf,Precision_df[classname[i]],label='Precision')
    plt.plot(sample_leaf,Recall_df[classname[i]],label='Recall')
    plt.plot(sample_leaf,FScore_df[classname[i]],label='FScore')
    plt.xticks(sample_leaf,['5','10','15','20','25'])
    plt.legend()
    plt.title("Evaluation for " + classname[i])
    plt.xlabel("min_leaf_size")
    plt.ylabel('Value')
    plt.show()
```







2 c). Limiting min_sample_leaf to 10

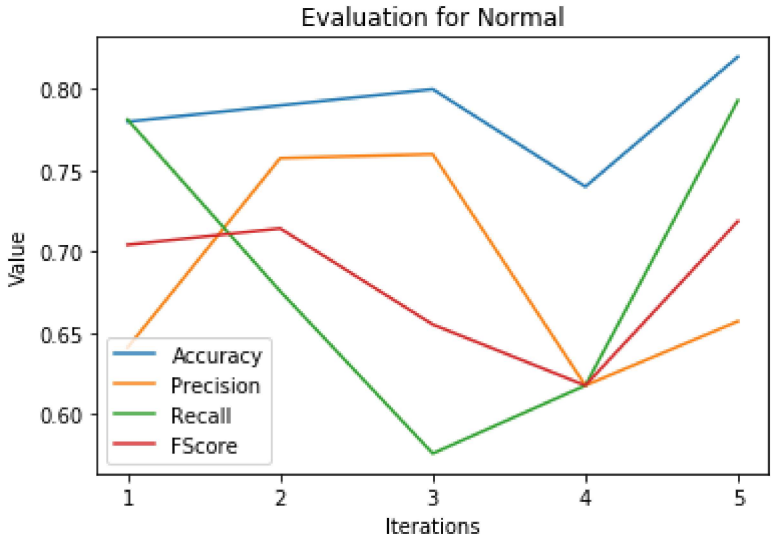
Execution Decision Tree for multiple iterations

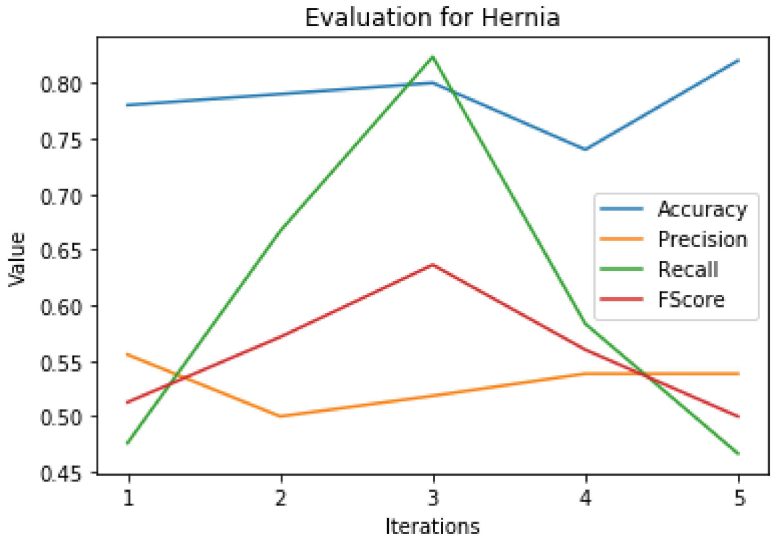
In [91]:
```
Precision = []
Recall = []
FScore = []
Accuracy = []
for i in range(5):
    data3 = shuffle(data3)
    X = data3.loc[:,'pelvic_incidence':'degree_spondylolisthesis']
    y = data3.loc[:,'class']
    X_train,X_test,y_train,y_test = X[:210] , X[210:], y[:210] , y[210:]
    decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state
 = 100, min_samples_leaf=10)
    fit =  decision_tree.fit(X_train,y_train)
    pred = fit.predict(X_test)
    acc = accuracy_score(y_test, pred, normalize=True, sample_weight=None)
    result = precision_recall_fscore_support(y_test, pred, average=None, label
s=['Normal', 'Hernia', 'Spondylolisthesis'])
    print("\nIteration:" + str(i+1) + "\n Accuracy \t" +str(acc) + "\n Precisi
on \t" + str(result[0])
                          +"\n Recall \t"  +str(result[1]) + "\n F1 Score \t"
+ str(result[2]))
    Precision.append(result[0])
    Recall.append(result[1])
    FScore.append(result[2])
    Accuracy.append(acc)
```

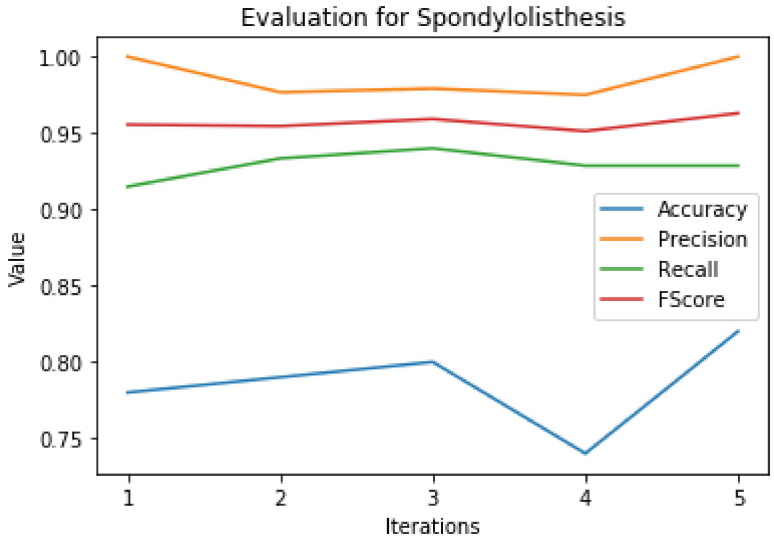
```
Iteration:1
 Accuracy       0.78
 Precision      [ 0.64102564  0.55555556  1.        ]
 Recall         [ 0.78125     0.47619048  0.91489362]
 F1 Score       [ 0.70422535  0.51282051  0.95555556]

Iteration:2
 Accuracy       0.79
 Precision      [ 0.75757576  0.5         0.97674419]
 Recall         [ 0.67567568  0.66666667  0.93333333]
 F1 Score       [ 0.71428571  0.57142857  0.95454545]

Iteration:3
 Accuracy       0.8
 Precision      [ 0.76        0.51851852  0.97916667]
 Recall         [ 0.57575758  0.82352941  0.94      ]
 F1 Score       [ 0.65517241  0.63636364  0.95918367]

Iteration:4
 Accuracy       0.74
 Precision      [ 0.61764706  0.53846154  0.975     ]
 Recall         [ 0.61764706  0.58333333  0.92857143]
 F1 Score       [ 0.61764706  0.56        0.95121951]

Iteration:5
 Accuracy       0.82
 Precision      [ 0.65714286  0.53846154  1.        ]
 Recall         [ 0.79310345  0.46666667  0.92857143]
 F1 Score       [ 0.71875     0.5         0.96296296]
```

Plotting Evaluations for different iterations

In [92]:
```
Precision_df = pd.DataFrame(Precision, columns=('Normal','Hernia','Spondylolis
thesis'))
Recall_df = pd.DataFrame(Recall, columns=('Normal','Hernia','Spondylolisthesi
s'))
FScore_df = pd.DataFrame(FScore, columns=('Normal','Hernia','Spondylolisthesi
s'))
classname = ['Normal','Hernia','Spondylolisthesis']

for i in range(3):
    plt.plot(sample_leaf,Accuracy,label='Accuracy')
    plt.plot(sample_leaf,Precision_df[classname[i]],label='Precision')
    plt.plot(sample_leaf,Recall_df[classname[i]],label='Recall')
    plt.plot(sample_leaf,FScore_df[classname[i]],label='FScore')
    plt.xticks(sample_leaf,['1','2','3','4','5'])
    plt.legend()
    plt.title("Evaluation for " + classname[i])
    plt.xlabel("Iterations")
    plt.ylabel('Value')
    plt.show()
```





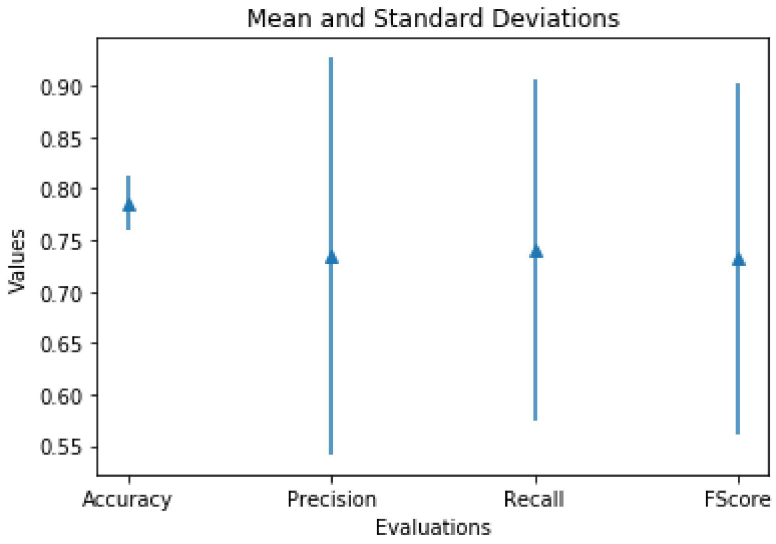


Calculating average and standard deviations of, Precision, Recall, Fscore and Accuracy

In [93]:
```
# average Accuracy, Precision, Recall and FScore
average = []
avg_accuracy = np.mean(Accuracy)
avg_precision = np.mean(Precision)
avg_recall = np.mean(Recall)
avg_fscore = np.mean(FScore)
average.append(avg_accuracy)
average.append(avg_precision)
average.append(avg_recall)
average.append(avg_fscore)

# standard deviation for Accuracy, Precision and Recall
stdev = []
std_accuracy = np.std(Accuracy)
std_precision = np.std(Precision)
std_recall = np.std(Recall)
std_fscore = np.std(FScore)
stdev.append(std_accuracy)
stdev.append(std_precision)
stdev.append(std_recall)
stdev.append(std_fscore)
```

Plotting average and standard deviations for Precision, Recall, Fscore and Accuracy

In [94]:
```
xax = ["Accuracy","Precision", "Recall", "FScore"]
step = [1,2,3,4]
plt.figsize = (30,15)
plt.errorbar(step, average, stdev, linestyle='None', marker='^')
plt.xlabel("Evaluations")
plt.ylabel("Values")
plt.title("Mean and Standard Deviations")
plt.xticks(step, xax)
plt.show()
```



2 d). Analysis

Comparing the graphs from 1c and 2c, it can be identified the measurements of accuracy, Precision, Recall and Fscore decreases for 2c. In 2c the decision tree predicts values for 3 classes while 1c predicts values for just 2 classes. On increasing the number of classes the performance of decision gradually decreases.

Answer 3

3 a). Importing data and separating features and label

In [32]:
```
data2 = pd.read_csv('C:/Users/rachi/Documents/UC Study Material/Sem 2/Intellig
ent Data Analysis/Assignments/Assignment 1/data2.csv')
X = data2.loc[:,'pelvic_incidence':'degree_spondylolisthesis']
y = data2.loc[:,'class']
```

Computing boundaries for each feature to create 4 equal bins

In [34]:
```
h1, boundary1 = np.histogram(X['pelvic_incidence'],bins=4)
h2, boundary2 = np.histogram(X['pelvic_tilt numeric'],bins=4)
h3, boundary3 = np.histogram(X['lumbar_lordosis_angle'],bins=4)
h4, boundary4 = np.histogram(X['sacral_slope'],bins=4)
h5, boundary5 = np.histogram(X['pelvic_radius'],bins=4)
h6, boundary6 = np.histogram(X['degree_spondylolisthesis'],bins=4)

boundary = [boundary1, boundary2, boundary3, boundary4, boundary5, boundary6]
```

In [36]:
```
colname = ['pelvic_incidence', 'pelvic_tilt numeric', 'lumbar_lordosis_angle',
 'sacral_slope',
               'pelvic_radius', 'degree_spondylolisthesis']
```

In [41]:
```
a=0
for j in range(6):
    for i in range(4):
        X[colname[j]][(X[colname[j]]>boundary[a][i]) & (X[colname[j]]<=boundar
y[a][i+1])] = i
    X[colname[j]][X[colname[j]]==boundary[a][0]] = 0
    a = a + 1

for i in range(6):
        print('\nBoundaries for - '+ colname[i])
        for j in range(4):
            print(str(boundary[i][j])+ ' - ' + str(boundary[i][j+1]) + ' : ' +
 str(j))
```

```
Boundaries for - pelvic_incidence
26.14792141 - 52.0694512075 : 0
52.0694512075 - 77.990981005 : 1
77.990981005 - 103.912510803 : 2
103.912510803 - 129.8340406 : 3

Boundaries for - pelvic_tilt numeric
-6.554948347 - 7.44175463975 : 0
7.44175463975 - 21.4384576265 : 1
21.4384576265 - 35.4351606132 : 2
35.4351606132 - 49.4318636 : 3

Boundaries for - lumbar_lordosis_angle
14.0 - 41.935596375 : 0
41.935596375 - 69.87119275 : 1
69.87119275 - 97.806789125 : 2
97.806789125 - 125.7423855 : 3

Boundaries for - sacral_slope
13.3669307 - 40.382589425 : 0
40.382589425 - 67.39824815 : 1
67.39824815 - 94.413906875 : 2
94.413906875 - 121.4295656 : 3

Boundaries for - pelvic_radius
70.08257486 - 93.32969127 : 0
93.32969127 - 116.57680768 : 1
116.57680768 - 139.82392409 : 2
139.82392409 - 163.0710405 : 3

Boundaries for - degree_spondylolisthesis
-11.05817866 - 96.34213653 : 0
96.34213653 - 203.74245172 : 1
203.74245172 - 311.14276691 : 2
311.14276691 - 418.5430821 : 3
```

In [40]:
```
X.head()
```

Out[40]:

	pelvic_incidence	pelvic_tilt numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degr
0	1.0	2.0	0.0	1.0	1.0	0.0
1	0.0	1.0	0.0	0.0	1.0	0.0
2	1.0	2.0	1.0	1.0	1.0	0.0
3	1.0	2.0	1.0	1.0	1.0	0.0
4	0.0	1.0	0.0	0.0	1.0	0.0

3 b). Combining the features and label

In [42]:
```
data4 = pd.concat([X, y], axis=1)
```

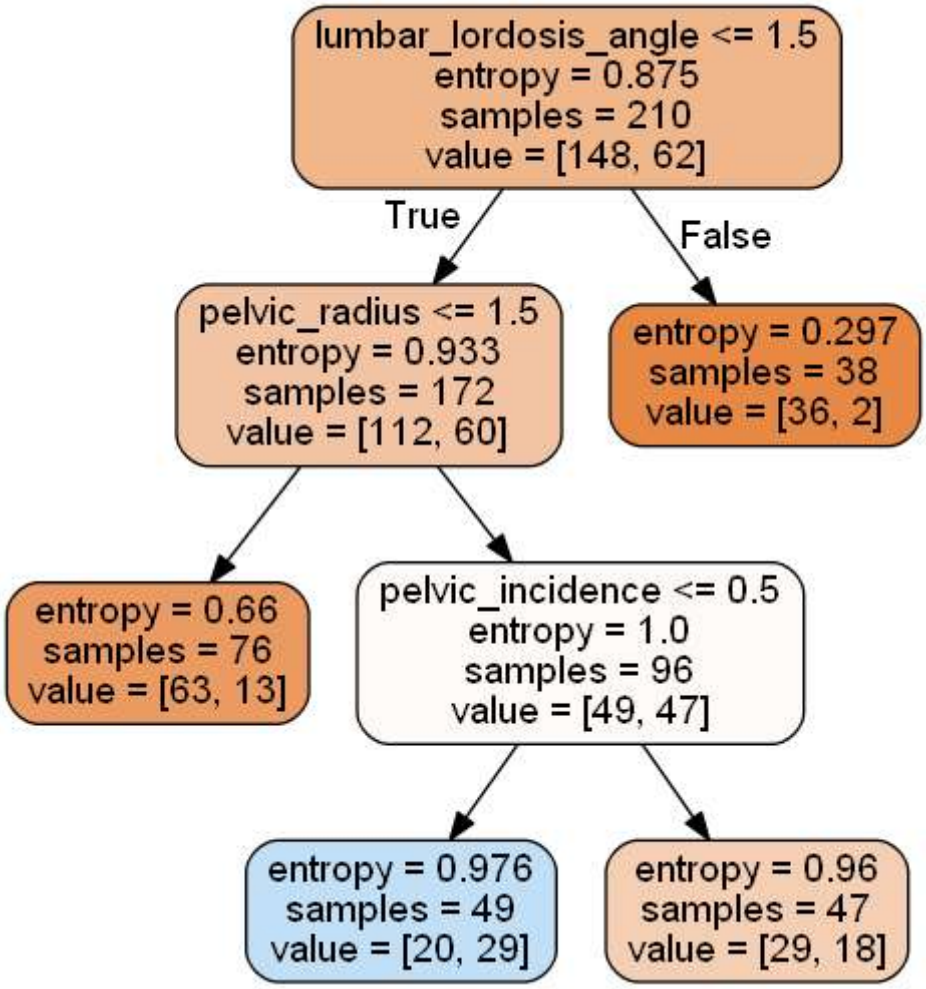
Shuffling data and separating training and testing data

In [44]:
```
data4 = shuffle(data4)
X = data4.loc[:,'pelvic_incidence':'degree_spondylolisthesis']
y = data4.loc[:,'class']
X_train,X_test,y_train,y_test = X[:210] , X[210:], y[:210] , y[210:]
```

Decision Tree with min_leaf_sample = 25

In [45]:
```
feats = ['pelvic_incidence','pelvic_tilt numeric','lumbar_lordosis_angle','sac
ral_slope','pelvic_radius','degree_spondylolisthesis']
decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state = 1
00, min_samples_leaf = 25)
fit = decision_tree.fit(X_train,y_train)     # Decision tree, Infromation gain
dot_data = tree.export_graphviz(fit, feature_names = feats, out_file=None, fil
led=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Out[45]:



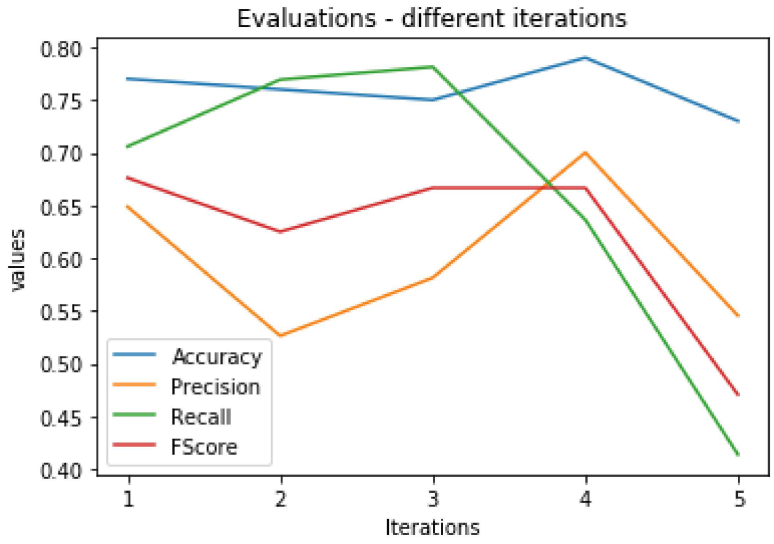
Evaluating for different iterations with min_leaf_sample = 10

In [46]:
```
Precision = []
Recall = []
FScore = []
Accuracy = []
for a in range(5):
    data4 = shuffle(data4)
    X = data4.loc[:,'pelvic_incidence':'degree_spondylolisthesis']
    y = data4.loc[:,'class']
    X_train,X_test,y_train,y_test = X[:210] , X[210:], y[:210] , y[210:]
    decision_tree = DecisionTreeClassifier(criterion = "entropy", random_state
 = 100, min_samples_leaf=10)
    fit = decision_tree.fit(X_train,y_train)
    pred = fit.predict(X_test)
    acc = accuracy_score(y_test, pred, normalize=True, sample_weight=None)
    evaluate = precision_recall_fscore_support(y_test, pred, pos_label='Norma
l', average='binary')
    print("\nIteration: " + str(a+1) + "\nAccuracy: " + str(acc) + "\nPrecisio
n: " + str(evaluate[0]) + "\nRecall: "
          + str(evaluate[1]) + "\nFscore: " + str(evaluate[2]) + "\n")

    Precision.append(evaluate[0])
    Recall.append(evaluate[1])
    FScore.append(evaluate[2])
    Accuracy.append(acc)
```

```
Iteration: 1
Accuracy: 0.77
Precision: 0.648648648649
Recall: 0.705882352941
Fscore: 0.676056338028


Iteration: 2
Accuracy: 0.76
Precision: 0.526315789474
Recall: 0.769230769231
Fscore: 0.625


Iteration: 3
Accuracy: 0.75
Precision: 0.581395348837
Recall: 0.78125
Fscore: 0.666666666667


Iteration: 4
Accuracy: 0.79
Precision: 0.7
Recall: 0.636363636364
Fscore: 0.666666666667


Iteration: 5
Accuracy: 0.73
Precision: 0.545454545455
Recall: 0.413793103448
Fscore: 0.470588235294
```

Plotting the Evaluations

In [51]:
```
iterations = [1,2,3,4,5]
plt.plot(iterations, Accuracy,label='Accuracy')
plt.plot(iterations, Precision,label='Precision')
plt.plot(iterations, Recall,label='Recall')
plt.plot(iterations ,FScore,label='FScore')
plt.legend()
plt.xticks(iterations,['1','2','3','4','5'])
plt.title("Evaluations - different iterations")
plt.xlabel("Iterations")
plt.ylabel("values")

plt.show()
```

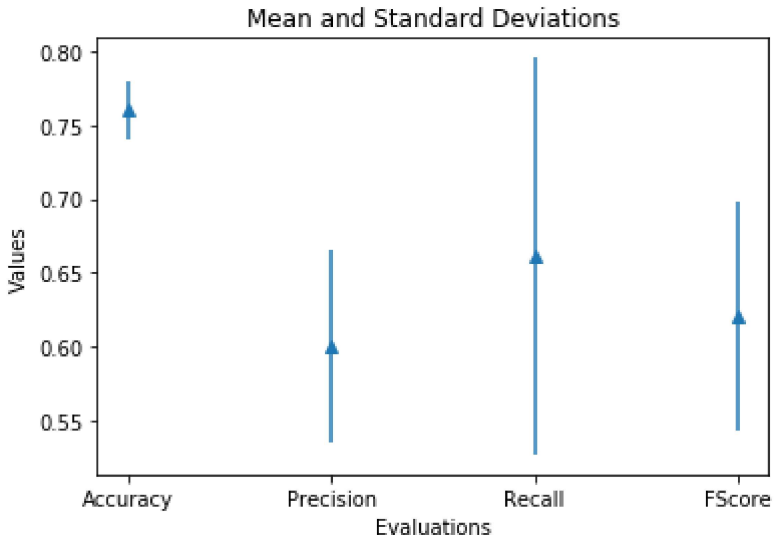


Computing average and standard deviation of Accuracy, Precision, Recall & Fscore

In [49]:
```
# average Accuracy, Precision, Recall and FScore
average = []
avg_accuracy = np.mean(Accuracy)
avg_precision = np.mean(Precision)
avg_recall = np.mean(Recall)
avg_fscore = np.mean(FScore)
average.append(avg_accuracy)
average.append(avg_precision)
average.append(avg_recall)
average.append(avg_fscore)

# standard deviation for Accuracy, Precision and Recall
stdev = []
std_accuracy = np.std(Accuracy)
std_precision = np.std(Precision)
std_recall = np.std(Recall)
std_fscore = np.std(FScore)
stdev.append(std_accuracy)
stdev.append(std_precision)
stdev.append(std_recall)
stdev.append(std_fscore)
```

Plotting average and standard deviations of Accuracy, Precision, Recall & Fscore

In [50]:
```
xax = ["Accuracy","Precision", "Recall", "FScore"]
step = [1,2,3,4]
plt.figsize = (30,15)
plt.errorbar(step, average, stdev, linestyle='None', marker='^')
plt.xlabel("Evaluations")
plt.ylabel("Values")
plt.title("Mean and Standard Deviations")
plt.xticks(step, xax)
plt.show()
```



3 c). Aanalysis

Comparing the graphs from 1c and 3b, the measurements of Precision, Recall, Fscore and Accuracy decreases for 3b. Since we are substituting the actual values random integer values, it will have a big impact on the performance of decision tree. Since the decision tree totally depends on the input training data and changing few values can drastically affect the predictions of decision tree. When we substitute actual values 0,1,2,3 instead of actual values we affect the integrity of data negatively and a huge decline in all evaluation metrices can be visualized. Hence the performance degrades for 3b as compared to 1c.