CS3481 Fundamentals of Data Science
Decision Tree Assignment

Hyder Ali (54028087)

(a)

First the complete data is partitioned to have the proportion of data records belonging to the different classes in both train and test sets to be similar to that of the original complete data set. This is done by setting the parameters in the split function, "train_test_split()" to random_state=10 to randomize the data and shuffle=True to shuffle the data as well before splitting. As it is split randomly into train and test sets, the proportion of different classes will be similar.

The impurity measure that I have used to construct the trees is the Gini impurity. The parameters that I have selected in the function, "DecisionTreeClassifier()" are mainly "criterion = gini" to define the criterion to base the splitting of each node which is the Gini impurity, "randome_state=10" to randomize the data by assigning a random seed number and "max_depth=3" such that the decision tree doesn't grow so large and become complicated which often happens in decision tree as it tries to learn and classify the data as perfect as possible based on the training set only. This will lead to over-fitting and the test results will not have high accuracy. The reason why I have selected the depth of the tree to be 3 is because I have tested different depths from 1 to 10 and even the complete decision tree on the unseen test data and compared their accuracy result. When the depth of the tree is 3, the accuracy is the highest. The tree, which this test was done was constructed using 70% of the complete data to be training set and 30% of the complete data to be the testing set. This is one of the most common baseline partitions thus it was logical to start off like this and see which depth of the tree is the most optimum. Now, we have the max_depth = 3 which is tested to be the most optimum in the 70-30 partition. Next, multiple trees are constructed using different partitions but with same max_depth=3. 70-30 partition means 70% of data is used for train and 30% for test.

(b)

The accuracy metric is used to measure the accuracy of the different tree classifiers constructed. The first tree constructed with 70-30 partition, has the accuracy of 0.838. The second tree with 50-50 partition, has the accuracy of 0.838, which is the same as the first tree. The third tree constructed using 30-70 partition has the accuracy of 0.801, which is understandably lower than previous trees as less data (30%) is used to train, thus the model doesn't have enough data to learn properly. I tested the accuracy of the third tree using different values of the max_depth with values from 1-10 and also with the complete binary tree. When max_depth = 4 or 6, the accuracy was the highest of about 0.8202. It is understandable as less data was used, a more complicated tree is needed to define the data so a slighter higher max_depth is needed to have better accuracy. Anyhow, with 70-30 and 50-50 partitions, the tree classifier still performed better in any case than with 30-70 partitions. I was interested if changing the max_depth of 2nd tree would increase its accuracy as well so I tested with values of max_depth from 1-10 and also with the entire tree and max_depth value of 6 is the most optimum for 2nd tree with accuracy of 0.877, which is much higher than 1st and 3rd trees. Thus this concludes that, when there is more data to train, the tree can generalize and predict the unseen data more properly and accurately on a fixed depth of the tree and if there's less data to train relatively, the depth of the decision tree would need to be increased such that the accuracy can increase. The table below concludes with the optimum max_depth values for the 3 trees.

	Optimum Max_depth	Partition	Accuracy
1st Tree	3	70-30	0.838
2nd Tree	6	50-50	0.877
3rd Tree	4	30-70	0.820

The plots of the 3 decision trees are given at the end of this report and can be viewed for references. Comparing the structures of the 3 trees with their optimum max_depth values, 1st tree is the simplest while 2nd tree is the most complicated. From the parent node, the class "NO" can be easily identified when the first condition is false while the other two classes are harder and are branched off from the true condition. This true condition branch needs more depth in all the 3 trees. This shows that the "DH" and "SL" classes are harder to distinguish from each other. The more depth needed for the trees are actually to distinguish these two classes. As depth grows, it also fits the irregularities and the structures of the tree can be seen to have some samples of "DH" and "SL" in each of their sides as the tree grows and tries to separate them.

(c)

As the 1st and 2nd trees perform the best in terms of the accuracy, these two trees are chosen. The confusion matrix is constructed to observe the classification performance associated with the different classes and to understand where the classifier confuses to distinguish between the different classes.

For 1st tree, the confusion matrix is constructed as below:

	Predicted			
Actual		DH	SL	NO
	DH	13	3	0
	SL	8	20	0
	NO	1	3	45

With respect to each class, performance evaluation is done with the accuracy metric.

For class DH, accuracy = 13/(13+3+0) = 0.8125

For class SL, accuracy = 20/(8+20+0) = 0.714

For class NO, accuracy = 45/(1+3+45) = 0.918

Thus, we can see that the classifier accurately predicts the NO class better than the other two. Thus in the 1st tree, DH and SL will likely be confused with each other. This is the same result as what we deduced earlier by observing the tree structure.

For 2nd tree, the confusion matrix is constructed as below:

	Predicted			
		DH	SL	NO

Actual	DH	24	4	1
	SL	8	39	2
	NO	1	3	73

With respect to each class, performance evaluation is done with the accuracy metric.

For class DH, accuracy = 24/(24+4+1) = 0.828

For class SL, accuracy = 39/(8+39+2) = 0.796

For class NO, accuracy = 73/(1+3+73) = 0.948

Thus, we can see that the classifier accurately predicts the NO class better than the other two. Thus in the 2nd tree, DH and SL will likely be confused with each other. This is the same result as what we deduced earlier by observing the tree structure.

In conclusion, in both 1st and 2nd trees, the DH and SL classes will likely be confused with each other as deduced earlier as well when observing the tree structure.

(d)

The selected confused class pairs are DH and SL. The leaf nodes, which classify to be DH, SL or NO class are labeled with DH, SL or NO class respectively. Please refer to the plots for each tree at the end of the report for references.

In both 1st and 2nd tree, the misclassification in both classes happens usually when there are irregularities in the data and the two classes can't be separated easily. The leaf nodes with Gini=0 means that it can classify the data up until this point perfectly based on the trained data. Thus, for leaf nodes with other values for Gini means that there would be some chance for misclassification as there is some impurity.

For 1st tree, the leaf node {Gini=0.444, class=SL} , could misclassify DH because the value of sacral_slope numeric has to be less 47.145 which is chosen based on the training set only and is not necessarily the case in test set. The leaf node {Gini=0.426, class=DH} could misclassify SL because the value of sacral_clope numeric has to be less than or equal to 28.135 which is chosen based on training set only and is not necessarily the case in test set. Both DH and SL are classified very decently at the leaf node {Gini=0.632, class=NO} to separate from NO class.

In the 2nd tree, the node {Gini=0.0, class=SL} could misclassify DH because pelvic_incidence numeric <= 63.69 in one of the upper node which I think is including a unnecessary irregularity of training data. Similarly, all the other leaf nodes, which have impurity and could most likely misclassify is due to a decision made in the upper nodes of the tree which made the leaf node contain impurity and depends on the decision which is based on training set only and might not be best for test set.

References

In [16]:
```
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import graphviz
```

In [17]:
```
data = pd.read_csv('../../Desktop/HW1/vertebral_column_data/column_
3C.dat', sep=' ', header=None)
data.columns = ['pelvic_incidence numeric', 'pelvic_tilt numeric',
'lumbar_lordosis_angle numeric', 'sacral_slope numeric', 'pelvic_ra
dius numeric','degree_spondylolisthesis numeric','class']
features = ['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lum
bar_lordosis_angle numeric', 'sacral_slope numeric', 'pelvic_radius
numeric','degree_spondylolisthesis numeric']
classes = ['disk hernia (DH)', 'spondylolisthesis (SL)', 'normal (N
O)']
```

In [18]:
```
X=data.iloc[:,0:6].values
Y=data.iloc[:,6].values
#print (len(data))
#print (data.shape)
```

In [ ]:
```

```

In [19]:
```
# Tree 1
```

In [20]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0
.3,random_state=10,shuffle=True)
```

In [21]:
```
c = tree.DecisionTreeClassifier(criterion = "gini", random_state =
10, max_depth=3)
c = c.fit(X_train, Y_train)
```

In [22]:
```
dot_data = tree.export_graphviz(decision_tree=c, out_file=None, fea
ture_names=features, class_names=classes, filled=True, rounded=True
, special_characters=True)
graph = graphviz.Source(dot_data)
graph.render('plot1', view=True)
```

Out[22]:
```
'plot1.pdf'
```

In [23]:
```
prediction = c.predict(X_test)
print(accuracy_score(Y_test, prediction))
```

```
0.8387096774193549
```

In [24]:
```
print(confusion_matrix(Y_test, prediction))
```

```
[[13  3  0]
 [ 8 20  0]
 [ 1  3 45]]
```

In [ ]:

In [25]:
```
# Tree 2
```

In [26]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0
.5,random_state=10,shuffle=True)
```

In [27]:
```
c = tree.DecisionTreeClassifier(criterion = "gini", random_state =
10, max_depth=6)
c = c.fit(X_train, Y_train)
```

In [28]:
```
dot_data = tree.export_graphviz(decision_tree=c, out_file=None, fea
ture_names=features, class_names=classes, filled=True, rounded=True
, special_characters=True)
graph = graphviz.Source(dot_data)
graph.render('plot2', view=True)
```

Out[28]:
```
'plot2.pdf'
```

In [29]:
```
prediction = c.predict(X_test)
print(accuracy_score(Y_test, prediction))
```

```
0.8774193548387097
```

In [30]:
```
print(confusion_matrix(Y_test, prediction))
```

```
[[24  4  1]
 [ 8 39  2]
 [ 1  3 73]]
```

In [ ]:

In [31]:
```
# Tree 3
```

In [32]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0
.7,random_state=10,shuffle=True)
```

In [33]:
```
c = tree.DecisionTreeClassifier(criterion = "gini", random_state = 
10, max_depth=4)
c = c.fit(X_train, Y_train)
```

In [34]:
```
dot_data = tree.export_graphviz(decision_tree=c, out_file=None, fea
ture_names=features, class_names=classes, filled=True, rounded=True
, special_characters=True)
graph = graphviz.Source(dot_data)
graph.render('plot3', view=True)
```

Out[34]:
```
'plot3.pdf'
```

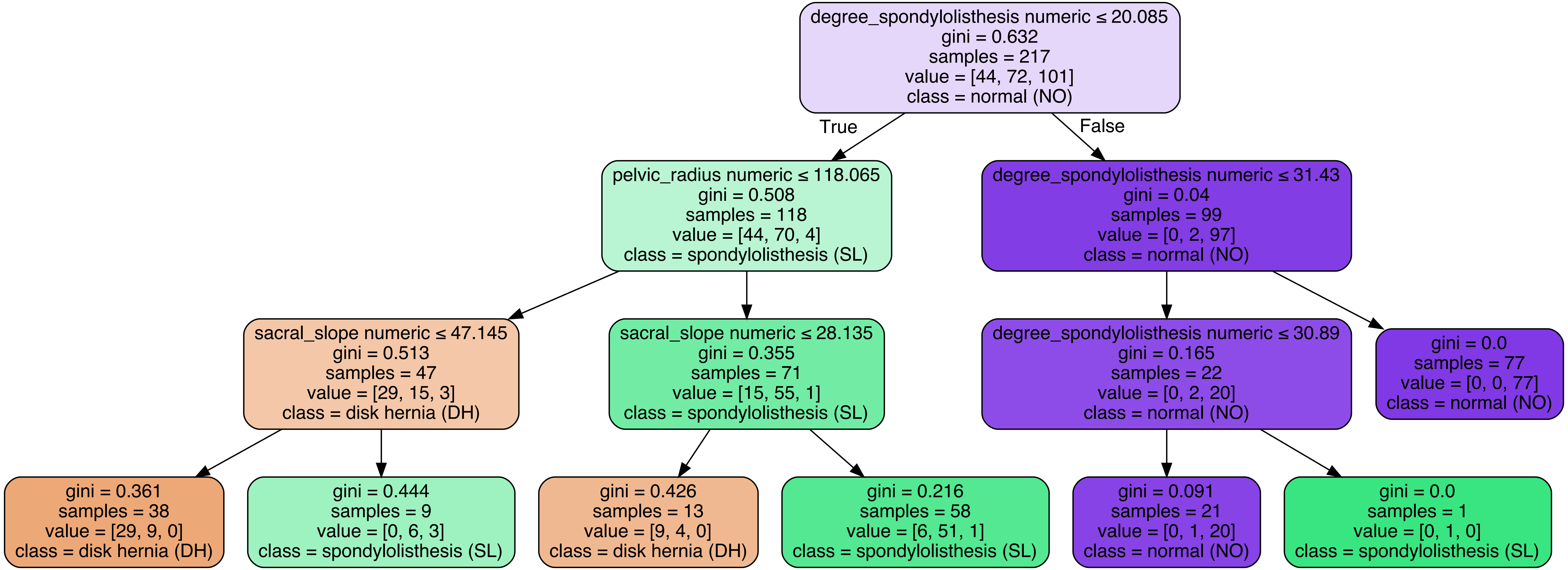
In [35]:
```
prediction = c.predict(X_test)
print(accuracy_score(Y_test, prediction))
```

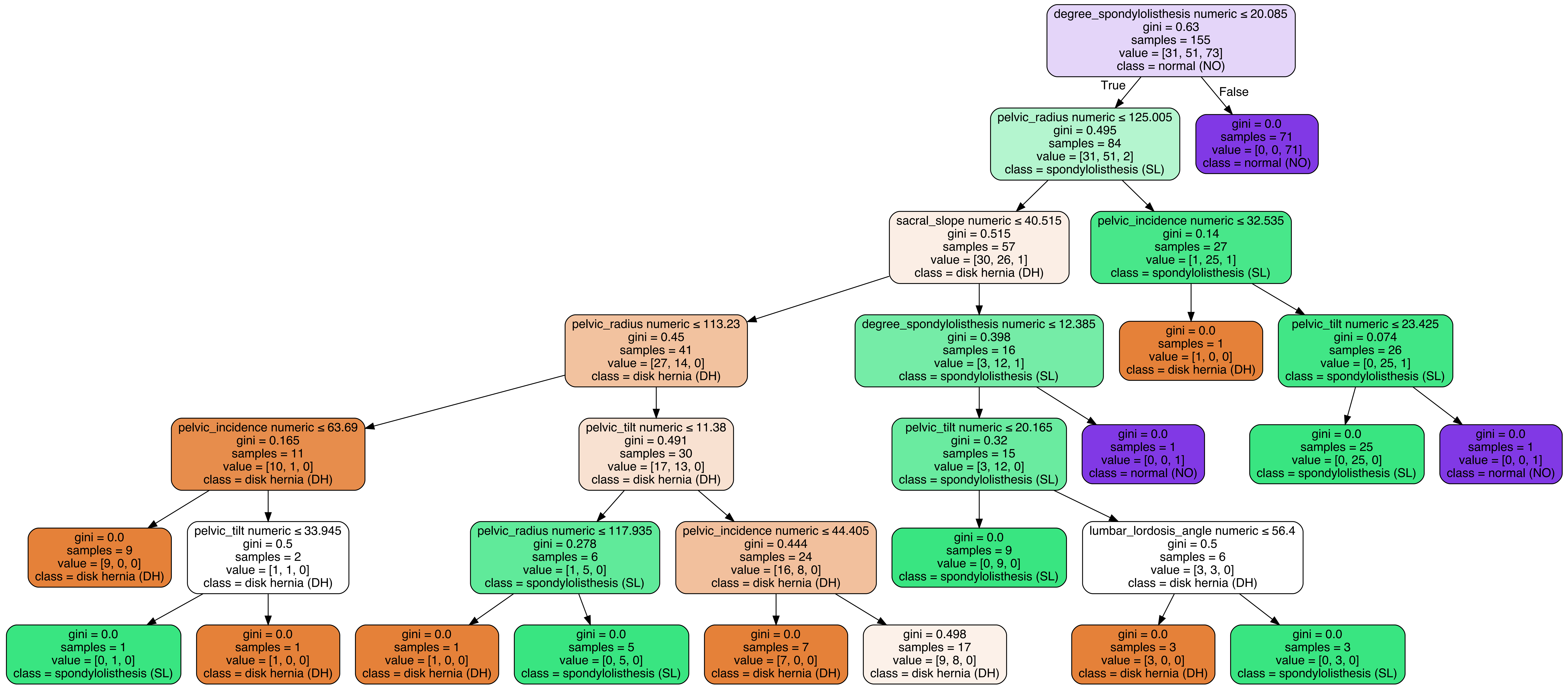
```
0.8202764976958525
```

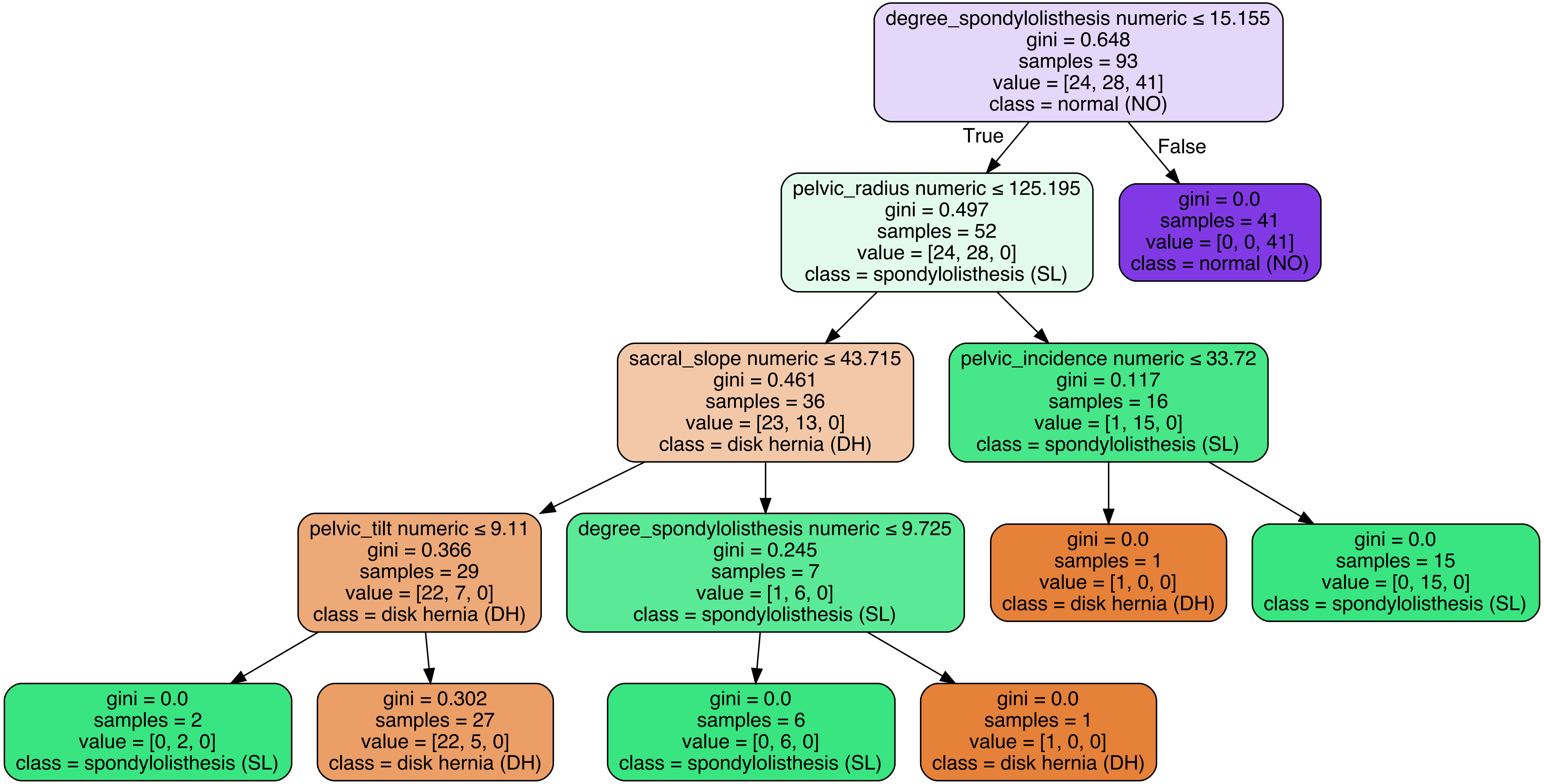
In [36]:
```
print(confusion_matrix(Y_test, prediction))
```

```
[[ 25  10   1]
 [ 20  49   3]
 [  4   1 104]]
```

In [ ]:


degree_spondylolisthesis numeric ≤ 20.085
gini = 0.632
samples = 217
value = [44, 72, 101]
class = normal (NO)
True
False
pelvic_radius numeric ≤ 118.065
gini = 0.508
samples = 118
value = [44, 70, 4]
class = spondylolisthesis (SL)
degree_spondylolisthesis numeric ≤ 31.43
gini = 0.04
samples = 99
value = [0, 2, 97]
class = normal (NO)
sacral_slope numeric ≤ 47.145
gini = 0.513
samples = 47
value = [29, 15, 3]
class = disk hernia (DH)
sacral_slope numeric ≤ 28.135
gini = 0.355
samples = 71
value = [15, 55, 1]
class = spondylolisthesis (SL)
degree_spondylolisthesis numeric ≤ 30.89
gini = 0.165
samples = 22
value = [0, 2, 20]
class = normal (NO)
gini = 0.0
samples = 77
value = [0, 0, 77]
class = normal (NO)
gini = 0.361
samples = 38
value = [29, 9, 0]
class = disk hernia (DH)
gini = 0.444
samples = 9
value = [0, 6, 3]
class = spondylolisthesis (SL)
gini = 0.426
samples = 13
value = [9, 4, 0]
class = disk hernia (DH)
gini = 0.216
samples = 58
value = [6, 51, 1]
class = spondylolisthesis (SL)
gini = 0.091
samples = 21
value = [0, 1, 20]
class = normal (NO)
gini = 0.0
samples = 1
value = [0, 1, 0]
class = spondylolisthesis (SL)


degree_spondylolisthesis numeric ≤ 20.085
gini = 0.63
samples = 155
value = [31, 51, 73]
class = normal (NO)
True
False
pelvic_radius numeric ≤ 125.005
gini = 0.495
samples = 84
value = [31, 51, 2]
class = spondylolisthesis (SL)
gini = 0.0
samples = 71
value = [0, 0, 71]
class = normal (NO)
sacral_slope numeric ≤ 40.515
gini = 0.515
samples = 57
value = [30, 26, 1]
class = disk hernia (DH)
pelvic_incidence numeric ≤ 32.535
gini = 0.14
samples = 27
value = [1, 25, 1]
class = spondylolisthesis (SL)
pelvic_radius numeric ≤ 113.23
gini = 0.45
samples = 41
value = [27, 14, 0]
class = disk hernia (DH)
degree_spondylolisthesis numeric ≤ 12.385
gini = 0.398
samples = 16
value = [3, 12, 1]
class = spondylolisthesis (SL)
gini = 0.0
samples = 1
value = [1, 0, 0]
class = disk hernia (DH)
pelvic_tilt numeric ≤ 23.425
gini = 0.074
samples = 26
value = [0, 25, 1]
class = spondylolisthesis (SL)
pelvic_incidence numeric ≤ 63.69
gini = 0.165
samples = 11
value = [10, 1, 0]
class = disk hernia (DH)
pelvic_tilt numeric ≤ 11.38
gini = 0.491
samples = 30
value = [17, 13, 0]
class = disk hernia (DH)
pelvic_tilt numeric ≤ 20.165
gini = 0.32
samples = 15
value = [3, 12, 0]
class = spondylolisthesis (SL)
gini = 0.0
samples = 1
value = [0, 0, 1]
class = normal (NO)
gini = 0.0
samples = 25
value = [0, 25, 0]
class = spondylolisthesis (SL)
gini = 0.0
samples = 1
value = [0, 0, 1]
class = normal (NO)
gini = 0.0
samples = 9
value = [9, 0, 0]
class = disk hernia (DH)
pelvic_tilt numeric ≤ 33.945
gini = 0.5
samples = 2
value = [1, 1, 0]
class = disk hernia (DH)
pelvic_radius numeric ≤ 117.935
gini = 0.278
samples = 6
value = [1, 5, 0]
class = spondylolisthesis (SL)
pelvic_incidence numeric ≤ 44.405
gini = 0.444
samples = 24
value = [16, 8, 0]
class = disk hernia (DH)
gini = 0.0
samples = 9
value = [0, 9, 0]
class = spondylolisthesis (SL)
lumbar_lordosis_angle numeric ≤ 56.4
gini = 0.5
samples = 6
value = [3, 3, 0]
class = disk hernia (DH)
gini = 0.0
samples = 1
value = [0, 1, 0]
class = spondylolisthesis (SL)
gini = 0.0
samples = 1
value = [1, 0, 0]
class = disk hernia (DH)
gini = 0.0
samples = 1
value = [1, 0, 0]
class = disk hernia (DH)
gini = 0.0
samples = 5
value = [0, 5, 0]
class = spondylolisthesis (SL)
gini = 0.0
samples = 7
value = [7, 0, 0]
class = disk hernia (DH)
gini = 0.498
samples = 17
value = [9, 8, 0]
class = disk hernia (DH)
gini = 0.0
samples = 3
value = [3, 0, 0]
class = disk hernia (DH)
gini = 0.0
samples = 3
value = [0, 3, 0]
class = spondylolisthesis (SL)


degree_spondylolisthesis numeric ≤ 15.155
gini = 0.648
samples = 93
value = [24, 28, 41]
class = normal (NO)
True
False
pelvic_radius numeric ≤ 125.195
gini = 0.497
samples = 52
value = [24, 28, 0]
class = spondylolisthesis (SL)
gini = 0.0
samples = 41
value = [0, 0, 41]
class = normal (NO)
sacral_slope numeric ≤ 43.715
gini = 0.461
samples = 36
value = [23, 13, 0]
class = disk hernia (DH)
pelvic_incidence numeric ≤ 33.72
gini = 0.117
samples = 16
value = [1, 15, 0]
class = spondylolisthesis (SL)
pelvic_tilt numeric ≤ 9.11
gini = 0.366
samples = 29
value = [22, 7, 0]
class = disk hernia (DH)
degree_spondylolisthesis numeric ≤ 9.725
gini = 0.245
samples = 7
value = [1, 6, 0]
class = spondylolisthesis (SL)
gini = 0.0
samples = 1
value = [1, 0, 0]
class = disk hernia (DH)
gini = 0.0
samples = 15
value = [0, 15, 0]
class = spondylolisthesis (SL)
gini = 0.0
samples = 2
value = [0, 2, 0]
class = spondylolisthesis (SL)
gini = 0.302
samples = 27
value = [22, 5, 0]
class = disk hernia (DH)
gini = 0.0
samples = 6
value = [0, 6, 0]
class = spondylolisthesis (SL)
gini = 0.0
samples = 1
value = [1, 0, 0]
class = disk hernia (DH)