In [590]:
```
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

# Assignment 2
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
```

In [591]:
```
data = pd.read_csv('../../Desktop/CS3481- Fundamentals of Data Scie
nce/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 [592]:
```
X=data.iloc[:,0:6].values
Y=data.iloc[:,6].values
#print (len(data))
#print (data.shape)
```

In [ ]:
```
```

In [593]:
```
# Random Forest
```

In [594]:
```
# (a)
```

In [595]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0
.3,random_state=10,shuffle=True)
```

In [596]:
```
RFclf = RandomForestClassifier(n_estimators=5)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8387096774193549
```

In [597]:
```
RFclf = RandomForestClassifier(n_estimators=10)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8817204301075269
```

In [598]:
```
RFclf = RandomForestClassifier(n_estimators=20)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8602150537634409
```

In [599]:
```
RFclf = RandomForestClassifier(n_estimators=30)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8602150537634409
```

In [600]:
```
RFclf = RandomForestClassifier(n_estimators=50)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8817204301075269
```

In [601]:
```
RFclf = RandomForestClassifier(n_estimators=100)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8924731182795699
```

In [ ]:
```

```

In [620]:
```
# (b)

# The best random forest is one with the most trees among all I tes
ted. --> 100 trees.
```

In [621]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0
.3,random_state=10,shuffle=True)
```

In [622]:
```
RFclf = RandomForestClassifier(n_estimators=100)
RFclf = RFclf.fit(X_train, Y_train)
print(RFclf.score(X_test,Y_test))
#RFprediction = RFclf.predict(X_test)
#print(accuracy_score(Y_test, RFprediction))
```

```
0.8817204301075269
```

In [646]:
```
# 11th component tree
DT1 = RFclf.estimators_[10]
DT1 = DT1.fit(X_train,Y_train)
print(DT1.score(X_test,Y_test))
```

```
0.8064516129032258
```

In [647]:
```
# 51th component tree
DT2 = RFclf.estimators_[50]
DT2 = DT2.fit(X_train,Y_train)
print(DT2.score(X_test,Y_test))
```

```
0.8172043010752689
```

In [666]:
```
# 91th component tree
DT3 = RFclf.estimators_[90]
DT3 = DT3.fit(X_train,Y_train)
print(DT3.score(X_test,Y_test))
```

```
0.7956989247311828
```

In [ ]:
```

```

In [631]:
```
# (c)
```

In [627]:
```
# for 11th tree -> n=10
```

In [632]:
```
print(features)
print(DT1.feature_importances_)    # the higher the value, the more
important the feature
```

```
['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lumbar_lordos
is_angle numeric', 'sacral_slope numeric', 'pelvic_radius numeric'
, 'degree_spondylolisthesis numeric']
[0.15054607 0.0223551  0.29845983 0.08816092 0.16113914 0.27933893
]
```

In [633]:
```
# for 51th tree -> n=50
```

In [634]:
```
print(features)
print(DT2.feature_importances_)
```

```
['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lumbar_lordos
is_angle numeric', 'sacral_slope numeric', 'pelvic_radius numeric'
, 'degree_spondylolisthesis numeric']
[0.0620108  0.06386526 0.09143613 0.09266998 0.06607688 0.62394094
]
```

In [635]:
```
# for 91th tree -> n=90
```

In [636]:
```
print(features)a
print(DT3.feature_importances_)
```

```
['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lumbar_lordos
is_angle numeric', 'sacral_slope numeric', 'pelvic_radius numeric'
, 'degree_spondylolisthesis numeric']
[0.30618497 0.11535975 0.07547124 0.0825032  0.19577095 0.22470989
]
```

In [637]:
```
# the feature important of the complete random forest
```

In [638]:
```
print(features)
print(RFclf.feature_importances_)
```

```
['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lumbar_lordos
is_angle numeric', 'sacral_slope numeric', 'pelvic_radius numeric'
, 'degree_spondylolisthesis numeric']
[0.1396279  0.09276069 0.11922486 0.11911471 0.12320436 0.40606747
]
```

In [ ]:
```
```

In [659]:
```
# (d) Naive Bayes Classifier
```

In [660]:
```
NBclf = GaussianNB()
```

In [661]:
```
NBclf = NBclf.fit(X_train, Y_train)
```

In [662]:
```
print(NBclf.score(X_test, Y_test))
#NBprediction = NBclf.predict(X_test)
#print(accuracy_score(Y_test, NBprediction))
```

```
0.8709677419354839
```

In [ ]:
```
```

In [ ]:
```
```

In [ ]:

In [77]:
```
# Notes:
```

In [ ]:

In [27]:
```
# accuracy_score returns --> no. of correct / total no. of samples,
when normalize=True which is by default
# so,that is same as when normalize=False and dividing the answer b
y no. of samples (len(data))
# usually, every classifier has its own score method to implement i
ts own performance metric, but in this case
# it turns our that random forest's score method is also using accu
racy_score method from the other library
# what i have seen, for random forest, it uses accuracy_score itsel
f in its .score method, the answer is the same
```

In [ ]:

In [ ]:
```
# in (b) , we need to train the each individual tree / component tr
ee of the random forest first. it will still
# be the same way the random forest trained, as all the parameters
are already set, such as max_features and
# the random state, stating how the features subset is selected and
how the data is randomized when training
# the decision tree. So its paramets is set, and even in random for
est, the different trees are just trained like
# that and the mode of the classification is slected --> highest vo
te / the accuracy is averaged too
```

In [ ]:

In [658]:
```
# in (c), the naive bayes classifier is used by implementing it as
the Gaussian naive bayes, because first of all,
# we are usin naive bayes because it's a classification problem -->
the target values/class labels are classes
# --> categorical values, thus naive bayes is used because it's a c
lassifier. Now, NB classifier can handle both
# categorical data or continuous numeric data. It treats categorica
l data by getting the prior probability by
# simpling counting, however for the numerica data, a distribution
function is used --> The type of distribution
# used is determined by the distribution existed in the type of the
data in the dataset. Most likely, in everyday
# cases, gaussian distribution can be good estimate so it's usually
used for numeric continuous values. Hense,
# we are using Gaussian Naive Bayes Classifer for our dataset in (c
)
```

In [ ]:

In [663]:
```
# Questions? (unsolved)
```

In [ ]:
```
# 1. What does random_state parameter does for decision tree algori
thm? I understand for random forest. But where is
# the randomness done in decision tree, and if not in features, as
there is option to not choose subset first.

# ANSWER --> I think i figured it out. The random_state for both de
cision tree and random forest is set to give
# randomness to any part of the algorithm which requires it. If you
set the specific seed, it can act as a way
# to give deterministic / constant / same values no matter how many
times you run the algorithm. This is good
# for testing using same result. As in random forest, max_features
is auto by default making the algorithm choose
# a subset of features RANDOMLY before choosing the best from the s
ubset. Also, as this algorithm chooses a subset
# of training samples too, it's very likely the randomnesss is also
applied to choosing the training samples first
# before applying to the different component trees which are indivi
dual decision trees. Now, decision trees algorithm
# by definition don't choose a subset of training samples first, so
we can assume, as we can't know for sure how
# the API / function is coded behind, that randomness is not applie
d here when applying the traning set to the model
# however it could be applied to decision tree when its paramter ma
x_feature is set to less than its maximum value
# then randomness of choosing the subset will be applied. This is t
he most logical explanation by me so far. So
# this means that, to train and test the individual trees of the fo
rest, I need to randomly select a subset myself
# too. So i can try this first!!! --> actually no way to know if ra
ndomness is applied in decision tree to the
# training data --> so it remains the only question of this assigme
nt --> ??????
```

In [ ]:

In [925]:
```
# Test
```

In [937]:
```
from random import seed
from random import randint
seed(100)
```

In [938]:
```
# 91th component tree
DT3 = RFclf.estimators_[90]
DT3 = DT3.fit(X_train,Y_train)
print(DT3.score(X_test,Y_test))
```

```
0.7956989247311828
```

In [939]:
```
DT3 = RFclf.estimators_[90]
XTrain = np.zeros((len(X_train),len(features)))
#YTrain = np.empty((len(Y_train),1), dtype="S10")
YTrain = ["" for yy in range(len(Y_train))]

for i in range (0,len(X_train)):
    row=randint(0,(len(X_train)-1))
    XTrain[i]=X_train[row]
    YTrain[i]=Y_train[row]

DT3 = DT3.fit(XTrain,YTrain)
print(DT3.score(X_test,Y_test))
```

```
0.7741935483870968
```

In [ ]:
```
```

In [929]:
```
XTrain
```

Out[929]:
```
array([[ 77.69,  21.38,  64.43,  56.31, 114.82,  26.93],
       [ 87.68,  20.37,  93.82,  67.31, 120.94,  76.73],
       [ 46.44,   8.4 ,  29.04,  38.05, 115.48,   2.05],
       ...,
       [ 54.5 ,   6.82,  47.  ,  47.68, 111.79,  -4.41],
       [ 38.05,   8.3 ,  26.24,  29.74, 123.8 ,   3.89],
       [ 84.59,  30.36,  65.48,  54.22, 108.01,  25.12]])
```

In [930]:
```
YTrain
```

Out[930]:
```
['SL',
 'SL',
 'DH',
 'NO',
 'NO',
 'NO',
 'SL',
 'SL',
 'NO',
 'NO',
 'DH',
 'DH',
 'NO',
 'DH',
 'SL',
 'SL',
```

```
'DH',
'DH',
'NO',
'SL',
'SL',
'NO',
'SL',
'NO',
'NO',
'NO',
'DH',
'SL',
'SL',
'DH',
'SL',
'NO',
'NO',
'NO',
'SL',
'DH',
'SL',
'SL',
'SL',
'SL',
'SL',
'NO',
'SL',
'SL',
'NO',
'NO',
'SL',
'NO',
'SL',
'NO',
'NO',
'SL',
'SL',
'SL',
'SL',
'DH',
'NO',
'NO',
'SL',
'SL',
'NO',
'DH',
'NO',
'NO',
'DH',
'NO',
'SL',
'DH',
'DH',
```

```
'NO',
'SL',
'SL',
'NO',
'SL',
'NO',
'NO',
'SL',
'DH',
'SL',
'SL',
'SL',
'SL',
'SL',
'DH',
'SL',
'SL',
'SL',
'DH',
'SL',
'SL',
'NO',
'SL',
'NO',
'SL',
'SL',
'SL',
'SL',
'SL',
'SL',
'DH',
'NO',
'DH',
'NO',
'SL',
'SL',
'SL',
'SL',
'NO',
'DH',
'DH',
'DH',
'SL',
'NO',
'NO',
'DH',
'NO',
'SL',
'SL',
'SL',
'NO',
'NO',
'SL',
'SL',
```

```
'NO',
'SL',
'SL',
'SL',
'SL',
'NO',
'DH',
'DH',
'SL',
'SL',
'NO',
'SL',
'NO',
'SL',
'SL',
'NO',
'DH',
'DH',
'SL',
'SL',
'SL',
'NO',
'DH',
'DH',
'SL',
'NO',
'SL',
'SL',
'SL',
'SL',
'SL',
'DH',
'SL',
'NO',
'SL',
'SL',
'NO',
'SL',
'SL',
'SL',
'SL',
'DH',
'NO',
'DH',
'NO',
'SL',
'DH',
'SL',
'SL',
'SL',
'SL',
'SL',
'NO',
```

```
 'SL',
 'DH',
 'SL',
 'NO',
 'SL',
 'NO',
 'SL',
 'SL',
 'SL',
 'SL',
 'SL',
 'SL',
 'DH',
 'SL',
 'SL',
 'SL',
 'SL',
 'SL',
 'SL',
 'DH',
 'DH',
 'NO',
 'NO',
 'NO',
 'SL',
 'SL',
 'SL',
 'NO',
 'SL',
 'SL',
 'SL',
 'NO',
 'NO',
 'NO',
 'SL',
 'DH',
 'SL',
 'NO',
 'SL',
 'NO',
 'NO',
 'SL']
```

In [931]:
```
len(YTrain)
```

Out[931]: 217

In [932]:
```
len(XTrain)
```

Out[932]: 217

In [ ]:
```

```

In [ ]:

In [ ]: