A. Construct random forest models using different numbers of component trees based on a specific training set/test set partition and analyze the resulting change in classification performance.

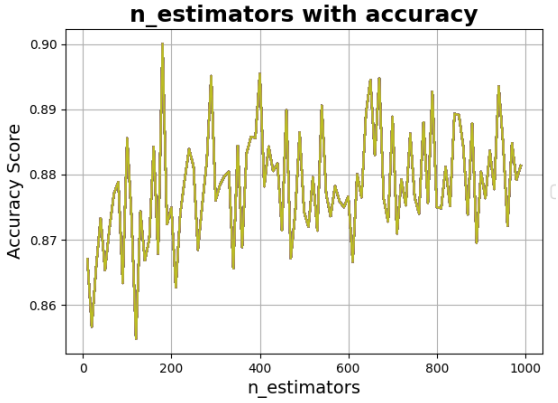
I have generated 10 ~ 100 (n_estimators) trees with the 80:20 (training data: testing data) partition. And I listed out the top 8 outcomes with the highest accuracy.

n_estimators	Average accuracy score in 10 loops
83	0.8387096774193549
58	0.8387096774193549
55	0. 8387096774193549
50	0.8387096774193549
41	0. 8387096774193549
40	0. 8387096774193549
22	0.8387096774193549
11	0.8387096774193549

However, the result does not fit my expectation, and I think it might be why the data distribution is not even. There are several significant attributes and less significant ones (found out from assignment 1).

```
>>>> RESULT >>>>
 [(0.8387096774193549, 83), (0.8387096774193549, 58), (0.8387096774193549, 55),
(0.8387096774193549, 50), (0.8387096774193549, 41), (0.8387096774193549, 40),
(0.8387096774193549, 22), (0.8387096774193549, 11) ..... ]
```

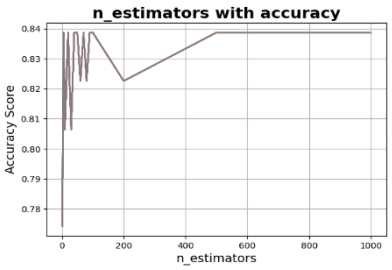
Besides, it is hard to analyze only from 10 – 100, which may contain too much outlier and cannot give out the data trend and indicate the relationship. Thus, I decided to plot the data with more significant estimators from 100 -> 1000 (+50).

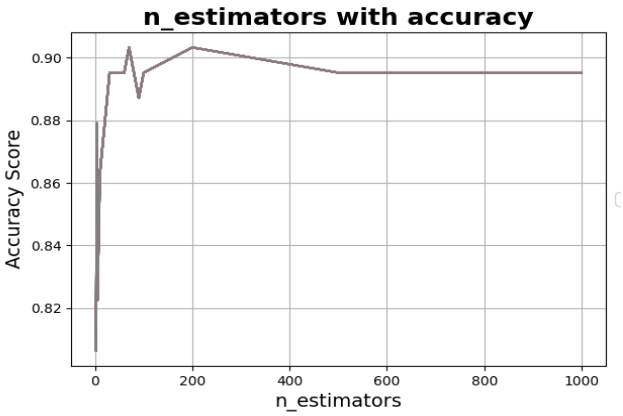


We can estimate that although there is a similar trend that we expected, the bias is still too large, and I decide to cross-validate with k_fold and get the mean score out of it.

running with k_fold ▶

▼ running with for loop

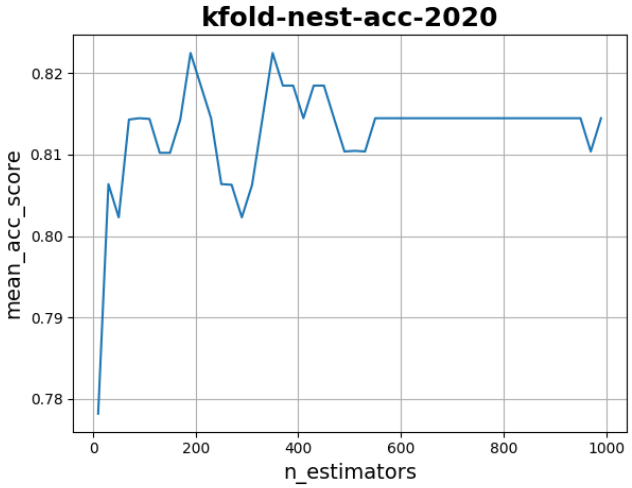




This graph clearly shows that the accuracy will increase along with the growth of the n_estimators. Another thing worth to be mentioned is that after the estimators of n, the accuracy seems to be the same, and I can suppose that it is the threshold of this dataset.

To get more accurate data, I run through the following parameters and get the graph to show the relation between n_estimators and mean_accuracy_score (k_fold).

```
classifier = RandomForestClassifier(n_estimators=i, random_state=0)
all_accuracies = cross_val_score(estimator=classifier, X=X_train, y=train_label, cv=5)
```



k-fold: 5

n_estimators: 1 - 1000

Train_test_split: 0.2

Random_state: 1

Self-define for loop (to get the mean of the mean accuracy score): 20

From this graph, we can also see a similar trend and found out that the more loops we run, the more accurate we will get. It also explained that the more trees we consider, the probability of outlier would reduce (more negligible effect) and the more generalized tree we will obtain.

Apart from the previously mention random forest classifier, which takes the default

parameter, I have used another attribute to get the best accuracy score among these parameters.

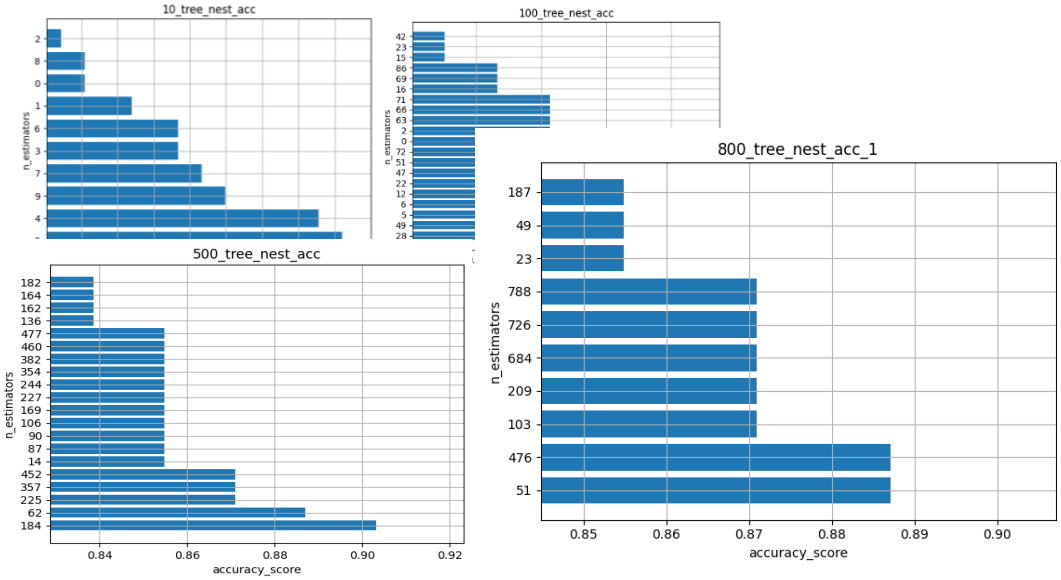
```
grid_param = {
        'n_estimators': [10, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000],
        'criterion': ['gini', 'entropy'],
        'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        'min_samples_leaf': [1, 2, 3, 4, 5, 6],
    }
gd_sr = GridSearchCV(estimator=classifier,
                     param_grid=grid_param,
                     scoring='accuracy',
                     cv=5,
                     n_jobs=-1)
```

The best parameter is`{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 6, 'n_estimators': 100}`

And the accuracy score is 0.8305306122448979. However, since the outcome seems not really to the expectation, I will only take this “best parameter” as a reference rather than a primary analyzing resource.

B. For the random forest model corresponding to the best classification performance, select different component decision trees in the model, and compare the classification performances of these trees with that of the original random forest model.

From question a, I observed that the best and stable accuracy happened during n_estimators = 500 ~ 1000. Even that the highest one occurs 190 and 380, it is not accepted since it’s unstable and the variance is too large. I have made another bar chart to go through all estimators in the selected n_estimators ([10, 100, 400, 500, 600, 800, 1000]) and find out the best accuracy among all of them.



From the above statistic, I have chosen the n_estimator: 800 with the following highest accuracy data. (index is to label the component (tree) in the forest)

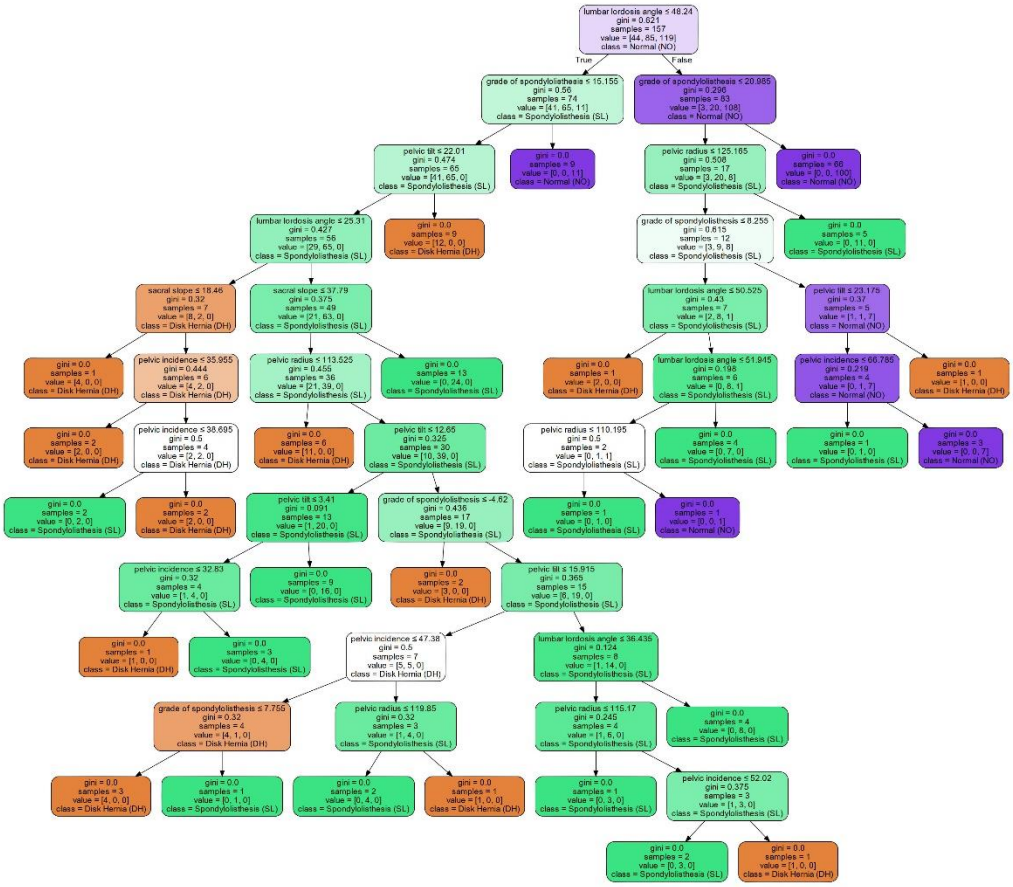
```
index:  51
accuracy_score:  0.8870967741935484
confusion_matrix:
[[ 7  4  0]
 [ 2 17  1]
 [ 0  0 31]]

index:  476
accuracy_score:  0.8870967741935484
confusion_matrix:
[[ 9  2  0]
 [ 4 16  0]
 [ 1  0 30]]

index:  103
accuracy_score:  0.8709677419354839
confusion_matrix:
[[ 9  2  0]
 [ 4 16  0]
 [ 0  2 29]]
```

◀ Top3 Accuracy decision tree (estimators)

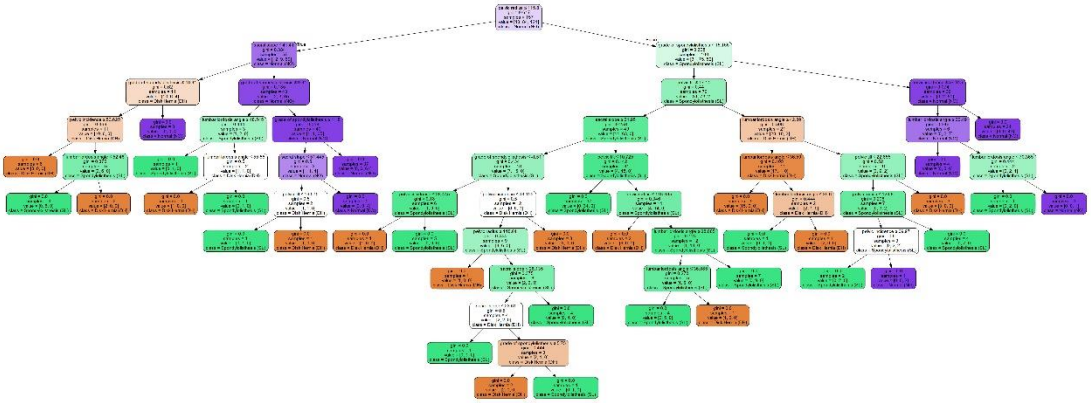
▼ original random forest model

```
accuracy_score:  0.8548387096774194
confusion_matrix:
[[ 8  3  0]
 [ 5 15  0]
 [ 0  1 30]]
feature_importance:
[0.12655697 0.08748487 0.13149767 0.13579338
0.12118764 0.39747947]
```

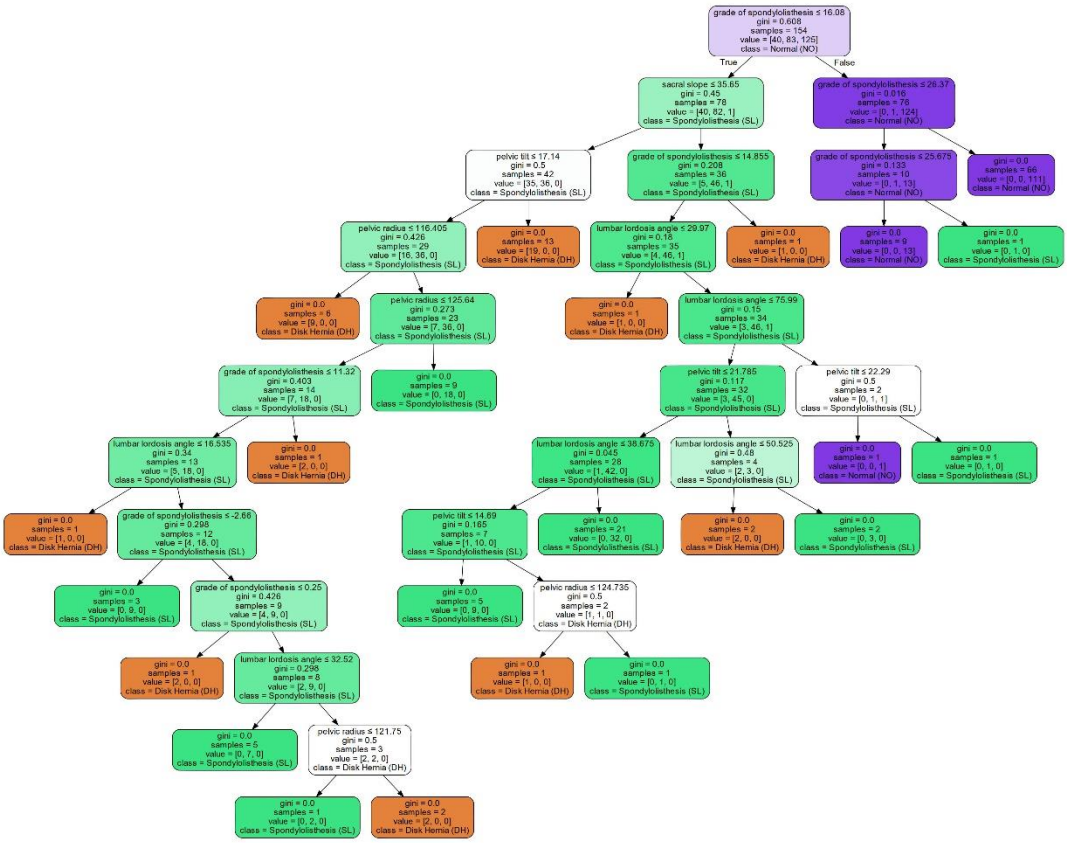
▼ decision tree of n_estimators: 800, index: 51



▼ decision tree of n_estimators: 800, index: 476



▼ decision tree of n_estimators: 800, index: 103



From the above three components' accuracy, we found out the exciting thing is that all of them have higher accuracy when predicting the testing data. And the reason is that random forest will construct a model by not choosing the "highest" accuracy tree but must go through bagging. It used the out-of-bag datasets to check how accurate the random forest is.

Another common feature among these "highest accuracy" tree is that the depth is way much higher, it means it will also occur the overfitting problem.

C. For a random forest classifier (or one of its component trees), the relative importance of the attributes can be measured through the feature_importances_ field of the classifier. For selected component trees in (b), compare their associated lists of relative attribute importance values.

The following table contains the impurity-based feature importance of previous selected components and classifier itself.

```
["pelvic incidence", "pelvic tilt", "lumbar lordosis angle", "sacral slope", "pelvic radius", "grade of spondylolisthesis"]
```

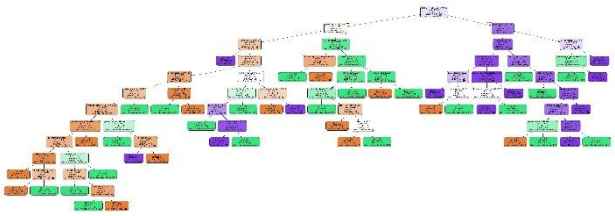
Original	[0.12655697 0.08748487 0.13149767 0.13579338 0.12118764 0.39747947]
Index: 51	[0.06051368 0.10468575 0.38269007 0.03074398 0.11456151 0.306805]
…476	[0.07209786 0.11641647 0.11940298 0.11897038 0.16910233 0.40400998]
…103	[0. 0.10896881 0.05067542 0.05980404 0.09998165 0.68057009]

From the observation, we can find out that the “grade of spondylolisthesis” has the highest relative importance value among all the attributes. Since the importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature, it means that the “grade of spondylolisthesis” usually occurs nearer the root of the tree for the original random forest model.

Even component 51 gets the “lumbar lordosis angle” as the highest importance attribute, but the other trees mainly have a similar result as the model. It can further prove out that the model is the synthesis of multiple trees.

Now I would like to select the worst two components in the forest, which only get the accuracy of 0.5 and 0.5645161290322581, respectively:

Original	[0.12655697 0.08748487 0.13149767 0.13579338 0.12118764 0.39747947]
Index: 151	[0.40212121 0.13323208 0.11558774 0.01798848 0.19160966 0.13946084]
…506	[0.18166334 0.04842365 0.33101918 0.13647426 0.18832766 0.1140919]



We can observe that the order of component 151’s relative feature importance value did not follow the original one's trend or demand. Which took “pelvic incidence,” “lumbar lordosis angle” as the most critical feature, and this also makes a huge decision tree consequently.

D. Construct a naïve Bayes classifier model based on our data set and compare the classification performance with that of the random forest model.

Based on Naïve Bayes' principle, the accuracy and confusion matrix would not change, and after running through these two classifiers with 0.2 (train: test) splits 100 times, I have the following table:

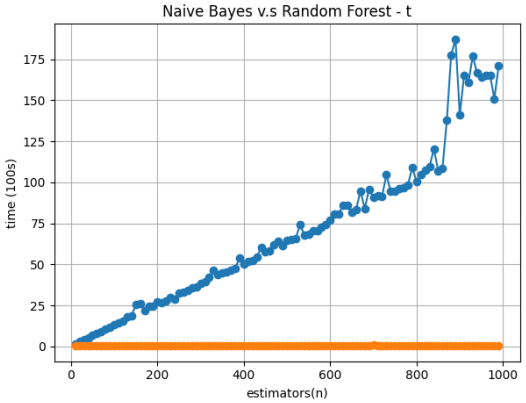
Naïve Bayes	Random Forest
0. 8064516129032258	0.8456451612903226
[[8 2 1] [5 14 1] [0 3 28]]	[[7 4 0] [5 15 0] [0 1 30]]

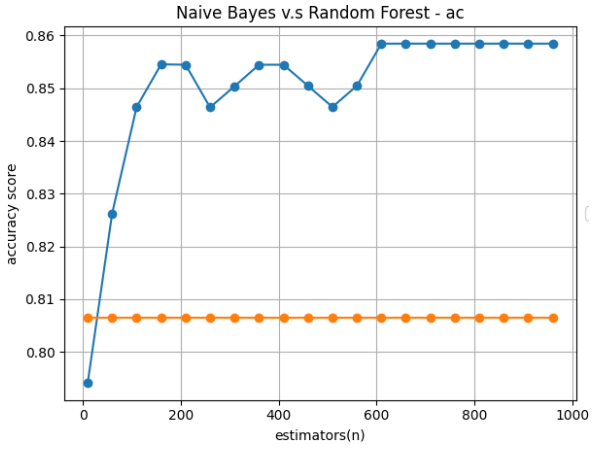
◀ ***Average(100 times)***

◀ ***Confusion Matrix***

The outcome is reasonable that the Naïve Bayes (NB) calculation method does not obtain any random figure as Random Forest (RF) does (bootstrapping). Thus, the score from NB with the same train/ test dataset will be constant. Another interesting is that the RF can always (mostly) get higher accuracy than NB since it takes a more prominent model size and time to construct the tree. However, it may also lead to the overfitting problem when using RF, and it is not adaptable when dealing with dynamic data. (not in our case, yet it is acceptable to apply to this assignment). Although NB cannot represent complex behavior, the time complexity will be much less than RF.

The following figures show the relationship of accuracy vs. estimators & time vs. estimators compared to NB & RF. And the figure clearly illustrates the above hypothesis.





From these graphs, we can find out that the NB always got the constant value and RF's time will increase linearly, and for the accuracy, we can obtain a similar curve as before but has higher accuracy than NB overall. For this assignment, we can use RF as the classifier for a higher accuracy model at dealing with the static data.