CS3481 Fundamental of Data Science

Assignment 2

SID: 56229799

Name: Chan Tsz Fung

Contents

(a)Construct random forest models using different numbers of component trees based on the default training set/test set partition, and analyze the resulting change in classification performance. (25%)

We have constructed a total of 5 random forests, by controlling the parameter n_estimator from 2 to 6:

```
In [106]: from sklearn.ensemble import RandomForestClassifier

clf6=RandomForestClassifier(n_estimators=6)
clf6=clf6.fit(forest_data,forest_target)
prediction6=clf6.predict(forest_data2)
print('\nprediction6 is \n'+str(prediction6))

clf5=RandomForestClassifier(n_estimators=5)
clf5=clf5.fit(forest_data,forest_target)
prediction5=clf5.predict(forest_data2)
print('\nprediction5 is \n'+str(prediction5))

clf4=RandomForestClassifier(n_estimators=4)
clf4=clf4.fit(forest_data,forest_target)
prediction4=clf4.predict(forest_data2)
print('\nprediction4 is \n'+str(prediction4))

clf3=RandomForestClassifier(n_estimators=3)
clf3=clf3.fit(forest_data,forest_target)
prediction3=clf3.predict(forest_data2)
print('\nprediction3 is \n'+str(prediction3))

clf2=RandomForestClassifier(n_estimators=2)
clf2=clf2.fit(forest_data,forest_target)
prediction2=clf2.predict(forest_data2)
print('\nprediction2 is \n'+str(prediction2))
```

```
prediction6 is
[2 0 0 2 0 2 1 2 0 2 0 3 2 1 3 0 3 0 0 0 3 0 0 0 0 0 0 2 0 2 0 3 0 0 2 2 3
 2 1 3 2 1 3 3 0 2 0 0 1 0 3 3 1 0 2 0 1 3 0 0 0 2 0 0 1 0 2 2 2 0 0 2 3 0
 1 0 0 0 0 0 2 0 0 1 2 0 0 3 2 0 3 2 0 0 0 2 2 0 2 2 0 1 0 0 0 0 3 3 3 0 3
 0 2 2 0 3 3 2 0 1 2 1 2 0 0 0 0 2 2 3 0 2 0 0 2 0 3 2 3 0 2 2 0 3 0 2 0 0
 0 1 1 2 3 2 3 2 3 2 1 2 0 2 2 0 0 0 2 2 0 1 0 3 2 0 0 1 2 0 2 2 2 2 0 2 2
 0 2 3 2 0 3 0 0 1 2 3 1 0 0 1 0 2 2 0 0 0 2 0 0 0 3 3 2 2 0 3 1 0 2 0 0 0
 0 1 0 2 0 0 0 0 0 3 0 0 0 0 3 2 2 3 3 0 0 3 2 0 2 0 0 2 2 0 2 2 2 0 0 0 0
 0 0 2 0 0 0 0 0 0 3 0 2 0 0 0 0 0 0 2 2 3 2 3 1 2 1 2 0 1 0 1 0 2 0 0 1 2
 0 0 0 2 2 0 1 1 0 0 3 0 0 1 0 0 0 0 0 1 0 1 0 2 3 2 0 0 0]

prediction5 is
[3 0 0 2 0 2 1 2 0 2 0 3 2 1 3 0 3 2 0 2 3 0 2 0 0 0 1 2 0 2 0 3 0 0 3 2 3
 2 1 3 2 1 3 3 0 3 0 0 1 0 2 3 1 1 2 2 1 3 1 1 0 2 0 0 1 0 2 2 2 0 0 2 3 0
 1 0 0 0 0 0 2 0 1 1 2 0 0 2 2 0 3 2 0 0 0 2 2 0 2 2 1 1 0 0 0 1 3 3 3 0 3
 0 2 2 0 3 3 2 0 1 2 1 2 0 0 0 0 2 2 3 2 2 0 0 2 0 3 2 0 0 2 2 0 3 0 2 0 0
 0 1 1 2 3 2 3 2 3 2 1 2 0 2 2 0 0 0 2 2 0 1 0 3 2 0 0 1 2 0 2 2 2 2 0 2 2
 0 2 3 2 0 3 0 2 1 2 3 1 0 0 1 0 2 2 0 0 0 2 0 0 0 3 3 2 2 1 2 1 2 2 0 0 2
 1 1 0 2 0 0 2 2 1 3 0 2 0 0 0 3 2 2 3 3 2 0 3 2 0 2 0 2 0 2 0 2 2 2 2 0 0 0
 0 0 2 0 0 0 0 0 0 3 0 2 0 0 0 0 0 0 2 2 3 2 2 1 2 1 2 1 1 0 1 0 2 0 0 1 0
 0 0 0 2 0 0 1 1 0 0 3 0 0 1 1 0 0 0 0 0 1 0 1 0 2 3 2 1 0 1]

prediction4 is
[3 0 0 2 0 2 1 2 0 2 0 3 2 1 3 0 3 2 0 2 3 0 0 0 0 0 0 2 0 2 0 3 0 0 3 2 3
 2 0 3 2 1 3 3 0 3 0 0 1 0 3 3 1 1 2 0 1 3 0 0 0 2 0 0 1 0 2 1 2 0 0 2 3 0
 1 0 0 0 0 0 2 2 0 1 2 0 0 3 2 0 3 2 0 0 0 2 2 0 2 2 0 1 0 0 0 0 3 3 3 0 3
 2 3 0 0 3 3 2 0 1 2 1 2 0 0 0 2 2 2 3 2 2 0 0 2 0 3 2 3 3 2 2 0 3 0 2 0 3
 0 1 1 2 3 2 3 2 3 2 1 2 0 2 2 0 0 0 2 2 0 0 0 3 2 0 0 1 2 0 2 2 2 2 0 2 2
 0 2 3 2 0 3 0 0 1 2 3 1 0 0 1 0 2 2 0 0 0 2 0 0 0 3 3 2 2 0 2 1 0 2 0 0 0
 0 1 0 2 0 0 0 0 0 3 0 0 0 0 0 3 2 2 3 3 2 0 3 0 0 2 0 0 0 2 0 2 2 2 2 0 0 0
 0 0 2 0 0 0 0 0 0 2 0 3 0 0 0 0 0 0 2 2 3 2 3 1 2 1 2 0 1 2 1 0 2 0 0 1 0
 0 0 0 0 0 1 1 0 0 3 0 1 1 0 0 0 0 0 1 0 1 0 2 3 2 0 0 0]

prediction3 is
[3 0 0 2 0 2 1 2 0 2 0 3 2 1 3 0 3 0 1 0 3 0 0 0 0 0 0 2 1 2 0 3 0 0 3 2 3
 2 1 3 2 1 3 3 0 3 0 0 1 0 2 3 1 0 2 0 1 3 1 0 0 2 0 0 1 0 2 1 2 0 0 2 3 0
 0 0 1 0 0 0 2 0 0 1 2 0 0 3 2 0 3 2 0 0 0 2 2 0 2 2 0 1 0 0 0 0 3 3 3 0 3
 0 3 2 0 3 3 2 0 1 2 1 2 0 0 1 0 0 2 3 0 2 1 0 2 0 3 2 0 0 2 2 0 3 0 2 0 0
 0 1 1 2 3 2 3 2 3 2 1 2 0 2 2 0 0 0 2 3 0 1 0 3 2 0 0 1 2 0 2 2 2 2 0 2 2
 0 2 2 2 0 3 1 0 1 2 3 1 0 0 1 0 2 2 0 0 0 2 0 0 0 3 3 2 2 0 2 1 0 2 0 0 0
 0 0 0 2 0 0 0 0 0 3 0 0 0 0 0 3 2 2 3 3 0 0 3 2 0 2 0 0 2 0 0 2 2 2 0 0 0
 0 0 2 0 0 0 0 0 0 2 0 3 0 0 0 0 0 0 2 2 3 2 3 1 2 1 2 0 1 2 1 0 2 0 0 1 2
 0 0 0 2 2 0 1 1 2 0 3 1 0 1 1 0 0 1 0 1 0 1 0 2 3 2 0 0 1]

prediction2 is
[2 0 0 2 0 2 0 0 1 2 0 3 0 0 3 0 2 0 1 2 3 0 0 0 0 0 0 0 1 1 0 3 0 0 2 2 3
 2 1 3 2 1 3 3 0 2 0 0 1 0 2 3 0 1 2 0 1 3 1 0 0 2 0 0 0 0 2 2 2 0 0 2 3 0
 1 0 0 0 0 0 2 0 0 1 2 0 0 3 2 0 3 2 0 0 0 2 2 0 2 2 0 1 0 0 0 0 3 3 2 0 3
 0 3 0 0 3 3 0 0 1 2 1 2 0 0 1 0 0 2 3 0 1 0 0 2 0 3 2 1 0 1 2 0 3 0 2 0 1
 0 1 0 2 3 1 3 2 3 0 1 2 0 1 2 0 0 0 1 2 0 1 0 3 2 0 0 1 2 0 2 2 2 2 0 2 2
 0 2 3 2 0 3 0 0 1 2 3 1 0 0 1 0 2 2 0 0 0 2 0 0 0 3 3 2 2 0 2 1 0 0 0 0 0
 0 1 0 1 0 0 0 0 0 3 0 0 0 0 0 3 2 2 3 3 2 0 3 1 0 2 0 0 0 2 0 2 2 2 0 0 0
 0 0 2 0 0 0 0 0 0 3 0 2 0 0 1 0 0 0 2 2 3 2 2 1 2 1 2 0 1 0 1 1 2 0 0 1 1
 0 0 0 2 0 0 1 1 0 0 3 1 0 1 1 0 0 1 0 1 0 1 0 2 3 2 0 0 1]
```

,

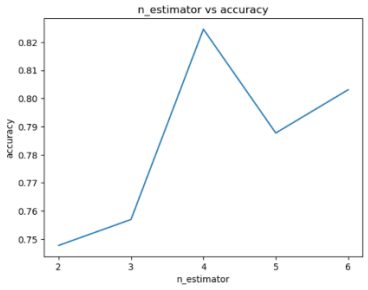
The confusion matrix and classification accuracy is shown as below:

In [107]:
```
##ACCURACY
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

print(accuracy_score(forest_target2,prediction6))
print(confusion_matrix(forest_target2,prediction6))
print('\n\n')
print(accuracy_score(forest_target2,prediction5))
print(confusion_matrix(forest_target2,prediction5))
print('\n\n')
print(accuracy_score(forest_target2,prediction4))
print(confusion_matrix(forest_target2,prediction4))
print('\n\n')
print(accuracy_score(forest_target2,prediction3))
print(confusion_matrix(forest_target2,prediction3))
print('\n\n')
print(accuracy_score(forest_target2,prediction2))
print(confusion_matrix(forest_target2,prediction2))
print('\n\n')
```

```
0.803076923076923
[[123   6   7   0]
 [ 11  27   0   0]
 [ 21   0  75   9]
 [  3   0   7  36]]



0.7876923076923077
[[113  12  11   0]
 [  5  32   1   0]
 [ 14   3  78  10]
 [  3   0  10  33]]



0.8246153846153846
[[127   6   3   0]
 [ 10  28   0   0]
 [ 16   0  76  13]
 [  1   0   8  37]]



0.7569230769230769
[[115  15   6   0]
 [ 11  27   0   0]
 [ 21   0  71  13]
 [  3   1   9  33]]



0.7476923076923077
[[121  13   2   0]
 [ 10  28   0   0]
 [ 26   7  63   9]
 [  1   3  11  31]]
```

We also plot the accuracy with respect to the number of estimator:

```
import matplotlib.pyplot as plt

n_estimator=[2,3,4,5,6]
accur=[0.7476923076923077,0.7569230769230769,0.8246153846153846,0.7876923076923077,0.803076923076923]

plt.plot(n_estimator, accur)
plt.title('n_estimator vs accuracy')
plt.xticks(list(range(2,max(n_estimator)+1)),[str(i) for i in range(2,max(n_estimator)+1)])
plt.xlabel('n_estimator')
plt.ylabel('accuracy')
plt.show()
```

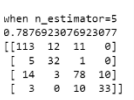


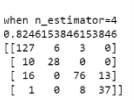
It is observable that the classification performance tends to improve when number of component trees increases. For example, the accuracy of classification is only 0.748 when number of component trees is two. The accuracy of classification improved gradually and become 0.7569 when n_estimator=3. The classification

accuracy reaches its peak when n_estimator=4. It dropped a little when n_estimator=5, increasing back when n_estimator =6. The overall trend is increasing. In other words, the accuracy positively correlates with number of estimators.

In general, we can conclude that the **more trees in a random forest, it is more likely that we could output an accurate prediction.**

The exceptional case is that the classification performance drops when number of estimators becomes 5. A one most important cause for this sudden drop is that number of "sugi" being wrongly classified as "Hinoki" or "Mixed" increased. Also, number of "other" being misclassified as "Sugi" or "Mixed" increased. We can reference that by checking row 1 and row4 of both confusion matrix:





When n_estimator=4, only 9 "Sugi" samples are being misclassified, while the misclassified number of samples raised to 23 when n_estimator=5. Similarly, the number of misclassified "other" samples is just 9 when n_estimator=4. It is increasesd to 13 when n_estimator=5.

The underlying reason for this phenomenon is that there are a lot more overfitting leaf nodes labelled as "Mixed" and "Hinoki" for the 5 trees constructed when n _estimator=5. Those trees constructed are not generalized well compared to those constructed when n_estimator=4.

In the following, we would try to analyze the cause of "Sugi being misclassified as Hinoki and Mixed" by looking into the leaf node, we would circle those overfitting leaf nodes which likely leads to the misclassification "Sugi being misclassified as Hinoki and Mixed" in red. (Criteria: Leaf node value is "Hinoki" or "Mixed", Parent is "Sugi", number of samples<10)

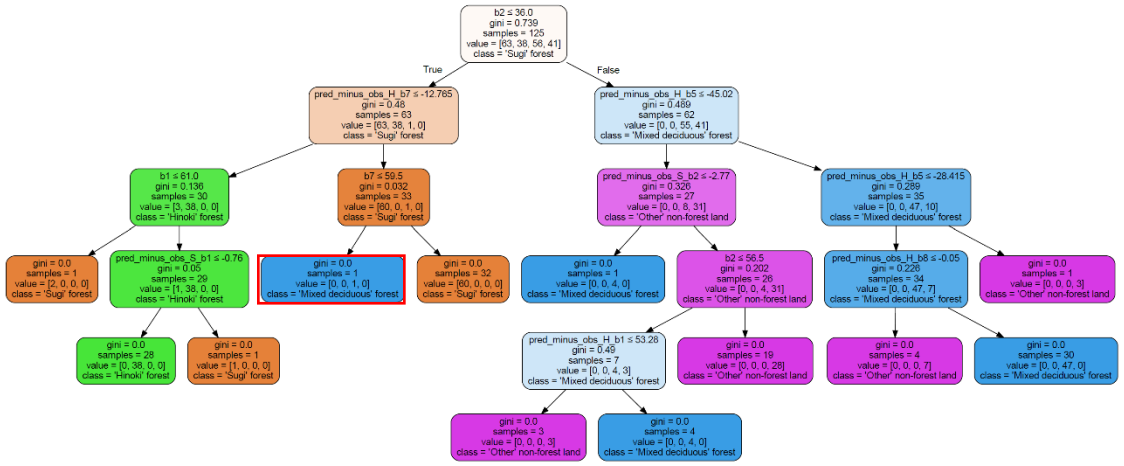


Figure 1 n_estimator=5, Tree1

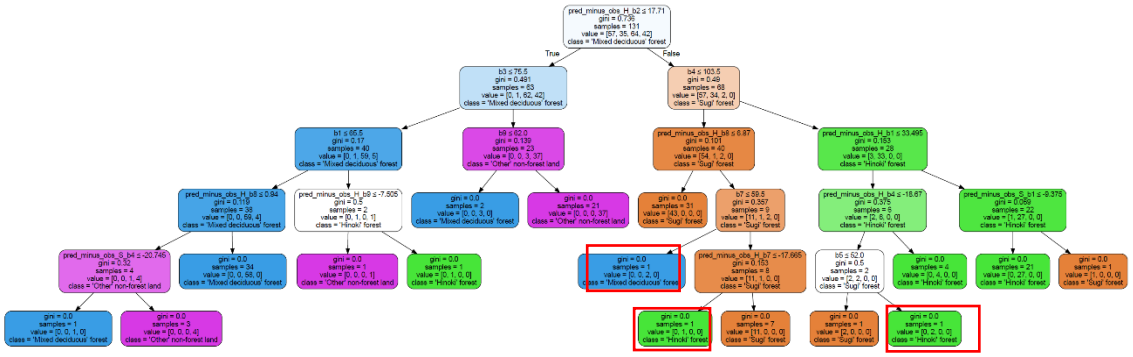


Figure 2 n_estimator=5, Tree2

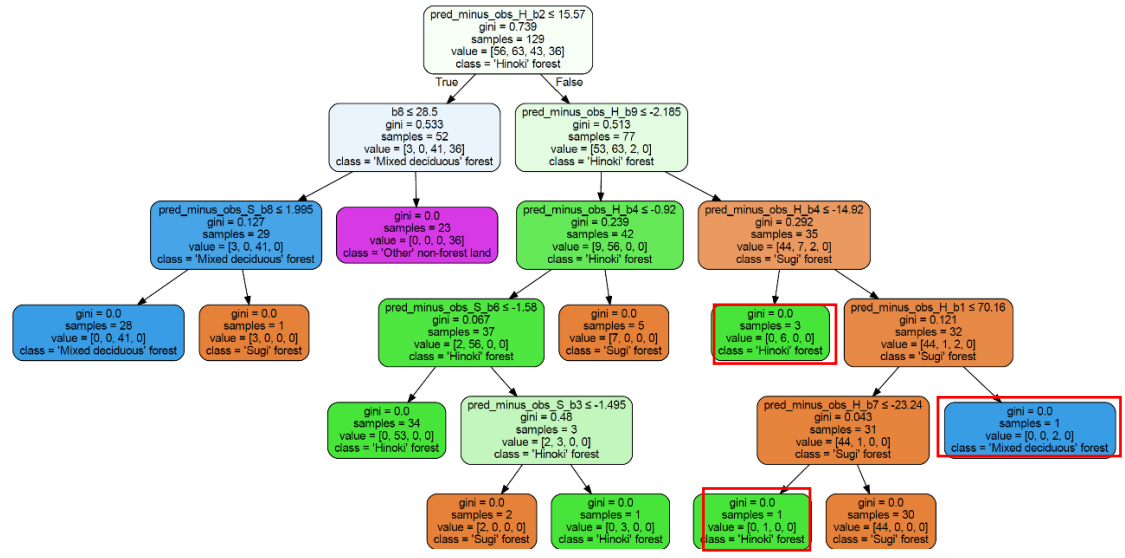


Figure 3 n_estimator=5, Tree3

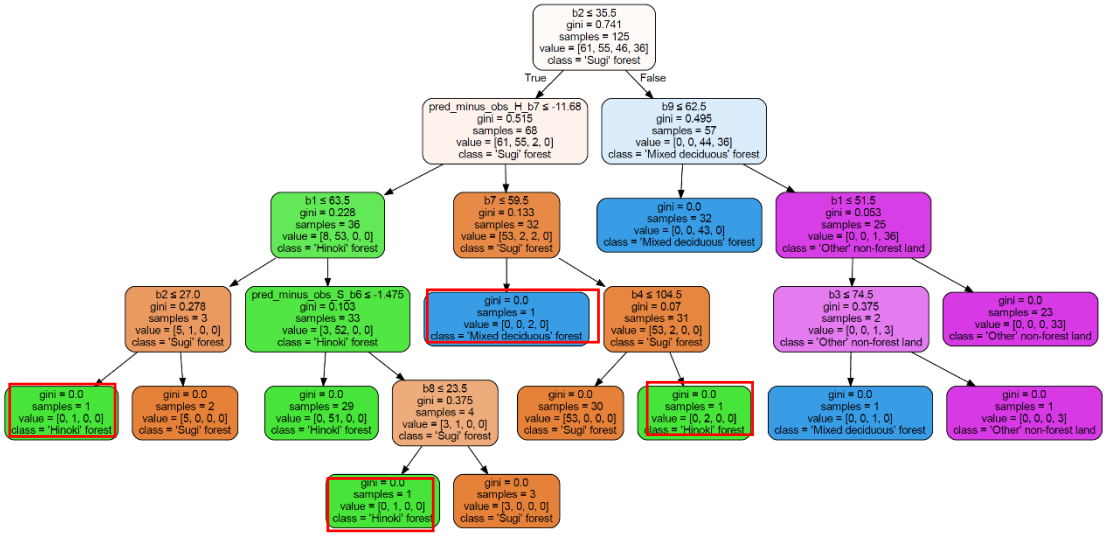


Figure 4 n_estimator=5, Tree4

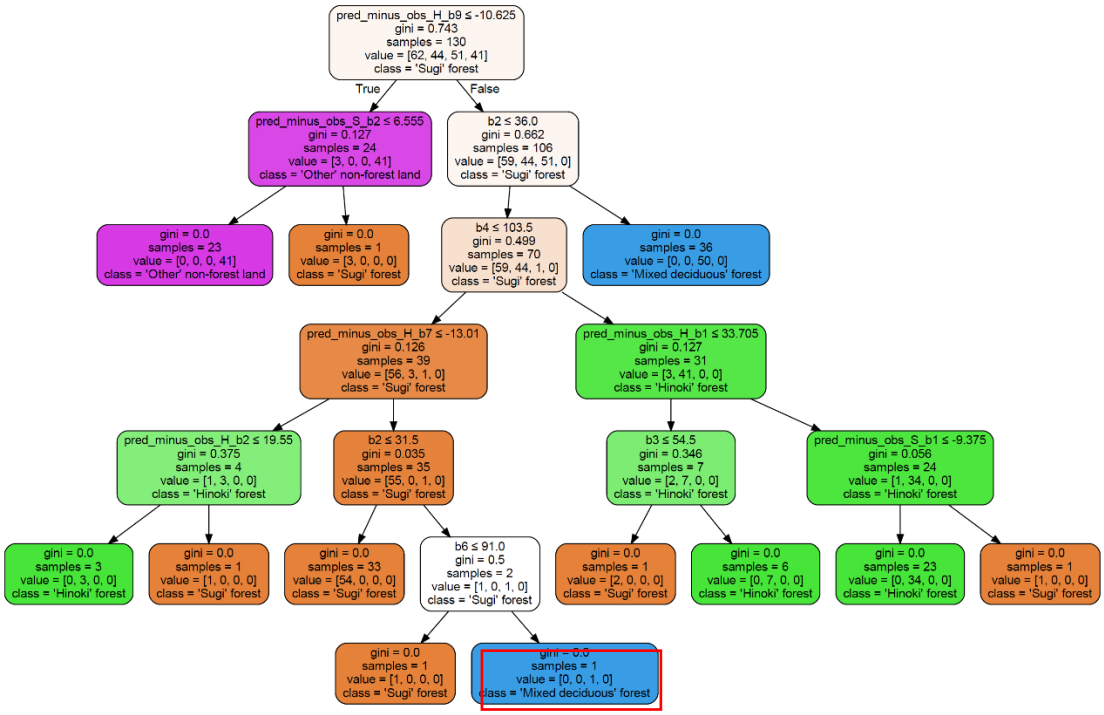


Figure 5 n_estimator=5, Tree5

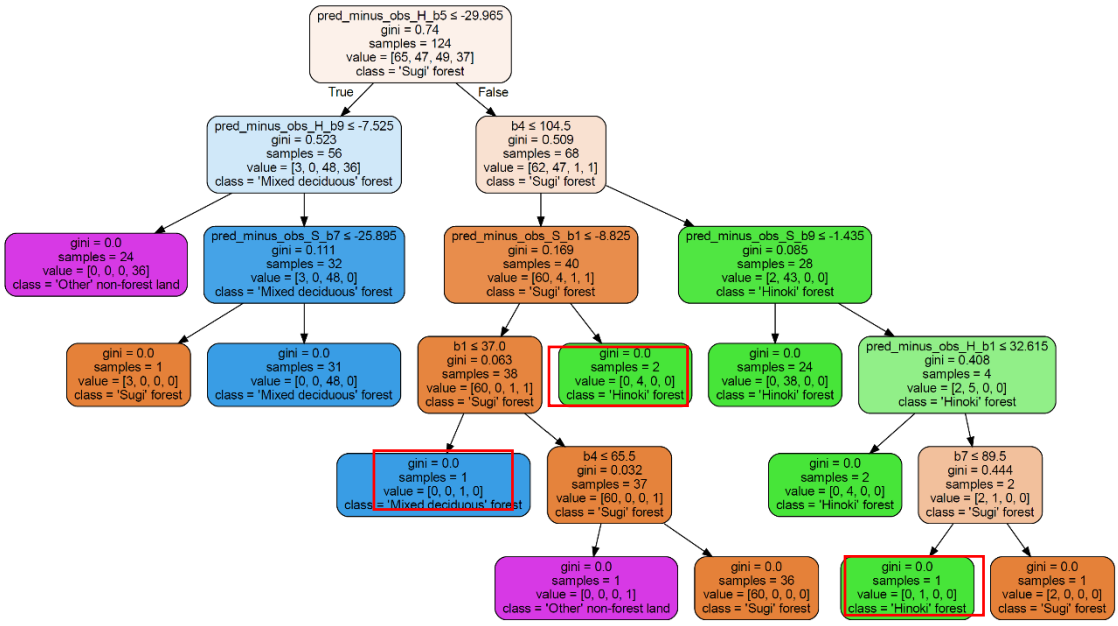


Figure 6 n_estimator=4, Tree1

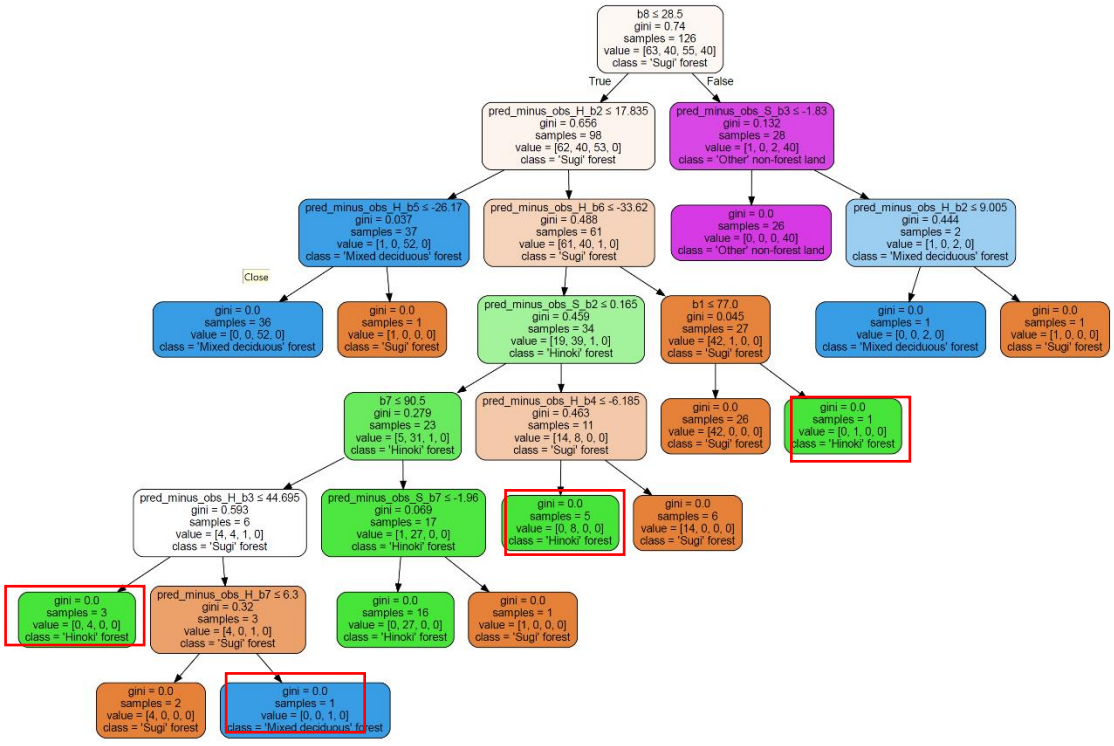


Figure 7 n_estimator=4, Tree2

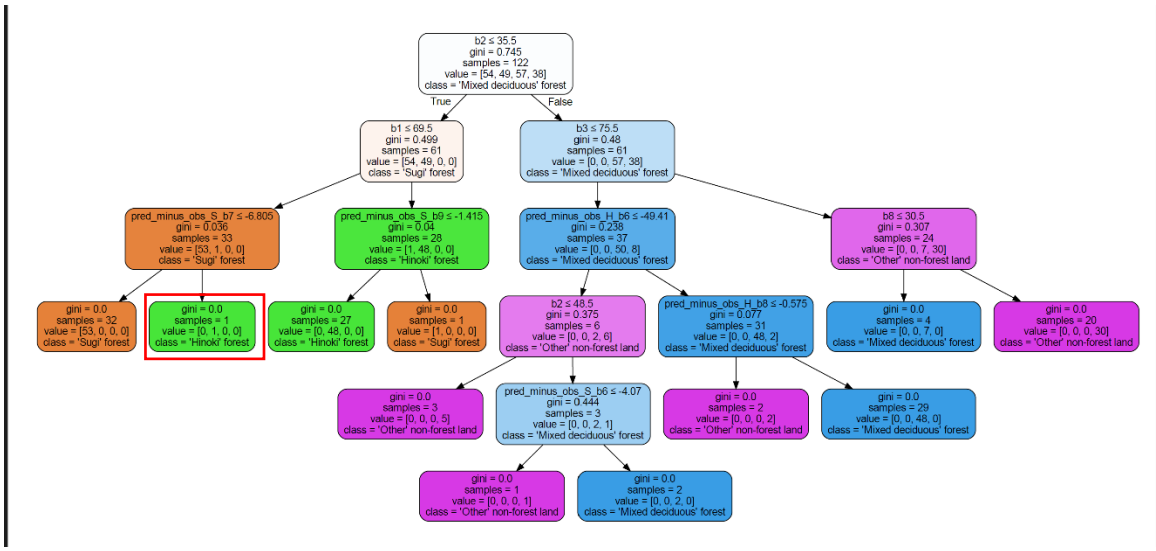


Figure 8 n_estimator=4, Tree3

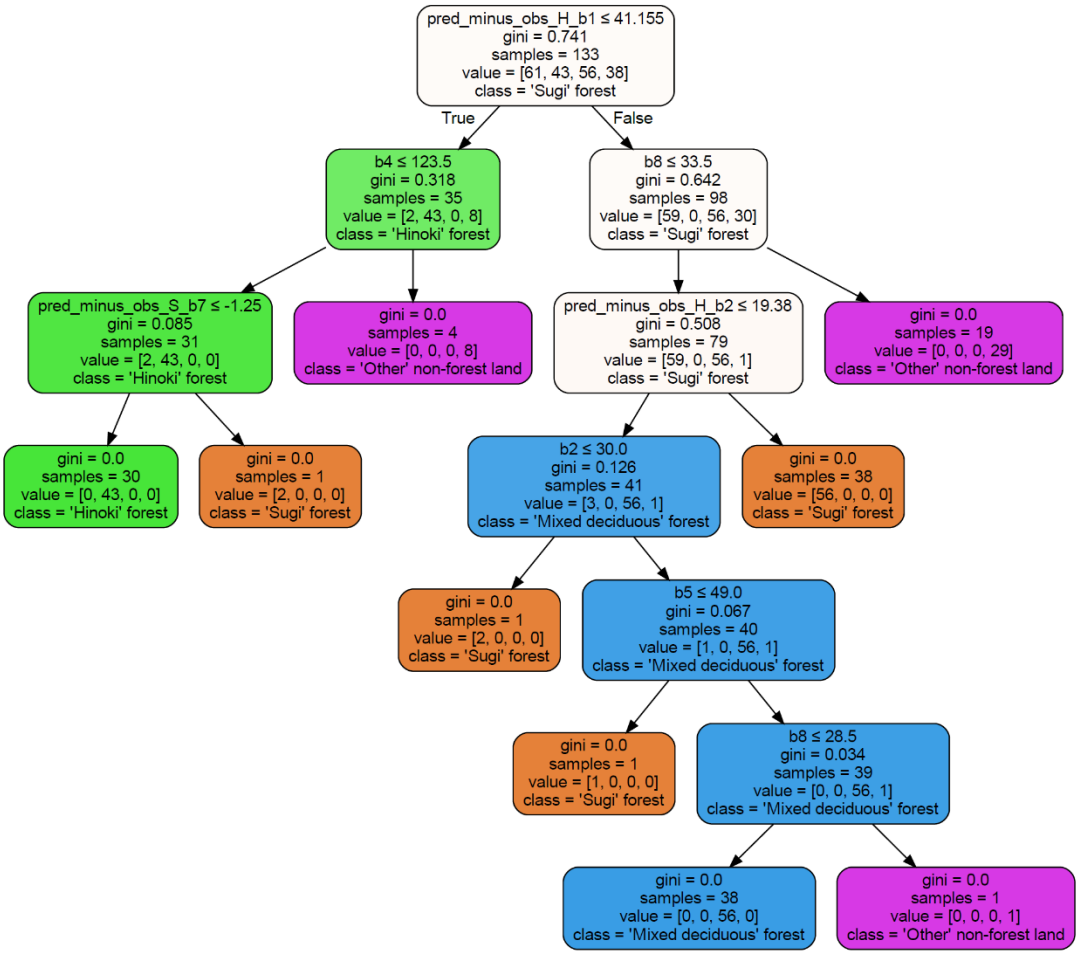


Figure 9 n_estimator=4, Tree4

Just by simple counting the leaf nodes who has few number of samples (<10) and labelled as Mixed or Hinoki, while having parents labelled as Sugi. We can see that:

Tree	No. of overfitting leaf nodes contribute to the misclassification
N_estimator=5; Tree 1	1
N_estimator=5; Tree 2	3
N_estimator=5; Tree 3	3
N_estimator=5; Tree 4	4
N_estimator=5; Tree 5	1
N_estimator=4; Tree 1	3
N_estimator=4; Tree 2	4
N_estimator=4 Tree 3	1
N_estimator=4; Tree 4	0

On average, trees constructed when n_estimator=5 have higher number of this kind of leaf nodes. ((1+3+3+4+1)/5) That is 2.4 overfitted leaf nodes per tree contributing to the misclassification. Meanwhile, the reason that n_estimator=4 achieves better generalization result is that only two ((3+4+1+0)/4) overfit leaf nodes per tree contributing to this kind of misclassification. This explains the dropped in classification accuracy when n_estimator is increased from 4 to 5.

In the following, we would try to continue analyzing confusion matrix for all the random forest. We would try to count the number of misclassified samples for each label to see which forest performs better in classifying particular class.

	n_est=6	n_est=5	n_est=4	n_est=3	n_est=2
Sugi	13	23	9	21	15
Hinoki	11	6	10	11	10
Mixed	30	27	29	34	42
Other	10	13	9	13	15
Total	64	69	57	79	82
Error Rate	0.197	0.212	0.175	0.243	0.252
Accuracy	0.803	0.788	0.825	0.757	0.748

For every type of classification, the forest with minimum misclassified samples is labelled as green and the forest with most misclassified samples is highlighted as red. We can see that forest with n_est=4 performs the best when it comes to classifying "Sugi" and "Other, while forest with n_est=5 performs the best when it comes to classifying "Hinoki" and "Mixed". However, forest with n_est=5 perform worst when it comes to classifying "Sugi", this can somehow explain why n_est=4 is better than n_est=5 for overall performance. Meanwhile, forest with n_est=2 perform worst when it comes to classifying "Mixed" and "Other". Forest with n_est=6 and forest with n_est=3 perform equally bad when it comes to classifying "Hinoki".

To summarize, we can somehow figure out that there are supplementary relationships among the forest with different number of estimators. There are good and bad classification cases for every constructed forest. The detailed relationship and how every forest is utilizing different attributes to achieve this result would be discussed in part C through the importance values.

(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. (25%)

The random forest model with the best classification performance consist of 4 component trees. The classification performance of these trees are shown below:

```
In [126]: print('when n_estimator=4')
          print(accuracy_score(forest_target2,prediction4))
          print(confusion_matrix(forest_target2,prediction4))
          print('\n For each component Tree :')
          clf4_1=clf4.estimators_[0]
          prediction4_1=clf4_1.predict(forest_data2)
          print(accuracy_score(forest_target2,prediction4_1))
          print(confusion_matrix(forest_target2,prediction4_1))
          print('\n')
          clf4_2=clf4.estimators_[1]
          prediction4_2=clf4_2.predict(forest_data2)
          print(accuracy_score(forest_target2,prediction4_2))
          print(confusion_matrix(forest_target2,prediction4_2))
          print('\n')
          clf4_3=clf4.estimators_[2]
          prediction4_3=clf4_3.predict(forest_data2)
          print(accuracy_score(forest_target2,prediction4_3))
          print(confusion_matrix(forest_target2,prediction4_3))
          print('\n')
          clf4_4=clf4.estimators_[3]
          prediction4_4=clf4_4.predict(forest_data2)
          print(accuracy_score(forest_target2,prediction4_4))
          print(confusion_matrix(forest_target2,prediction4_4))
          print('\n')

          when n_estimator=4
          0.8246153846153846
          [[127   6   3   0]
           [ 10  28   0   0]
           [ 16   0  76  13]
           [  1   0   8  37]]

           For each component Tree :
          0.7661538461538462
          [[111   9  15   1]
           [  7  26   5   0]
           [ 15   2  78  10]
           [  1   0  11  34]]


          0.7292307692307692
          [[104  18  14   0]
           [ 15  23   0   0]
           [ 11   4  74  16]
           [  2   0   8  36]]


          0.8123076923076923
          [[122   9   5   0]
           [  5  33   0   0]
           [ 20   0  74  11]
           [  2   0   9  35]]


          0.76
          [[111   6  19   0]
           [ 12  25   0   1]
           [ 12   4  76  13]
           [  0   4   7  35]]
```

In terms of accuracy score, the original random forest model achieves a better performance compared to every component tree. For the samples with truth value “Sugi” and “Other”, the classification performance of the random forest is the best compared to all of its component trees. To illustrate, random forest got 127 samples correctly classified as “Sugi”, while the number of correct “Sugi” classifications in all the other component trees are below this number (111,104,122,111). This shows that component trees have supplement each other for giving better classification results. Similar to “Other”, 37 samples are being correctly classified in the random forest, while the number of correct samples are lower than this number in every component trees(34,36,35,35).

Though overall classification performance is improved when using random forest, it is observed that not every type of classification is performed best when using random forest. There exist some component trees performing better in

classifying “Hinoki” and “Mixed”. For example, component tree 3 correctly classified 33 Hinoki samples, compared with only 28 samples in random forest. Also, component tree 1 correctly classified 78 Mixed samples, compared with only 76 samples in random forest. This shows the classification performance can get worsen when the tree with good performance are placed in large pool of randomly constructed trees. The improvement is never guaranteed, but relies on the performance of every constructed trees.

In the following, we would look further into those records which are correctly classified as Hinoki in tree 3, but fails to correctly classified in random forest. There are a total of (33-28=5samples). We are going to see how these five records go in every component trees that leads to misclassification.

```
import numpy as np
x=prediction4_1.size
forestTest= np.array(forest_data2)
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
print(forest_target_names)
for i in range(x):
    if prediction4_3[i]==1 and forest_target2[i]==1 and prediction4[i]!=1: #classify correct in tree 3 but not in forest
        print('record idx i is '+ str(i))
        print('For this Hinoki record,component tree 1 is classify as '+forest_target_names[int(prediction4_1[i])])
        print('For this Hinoki record,component tree 2 is classify as '+forest_target_names[int(prediction4_2[i])])
        print('For this Hinoki record,component tree 3 is classify as '+forest_target_names[int(prediction4_3[i])])
        print('For this Hinoki record,component tree 4 is classify as '+forest_target_names[int(prediction4_4[i])])
        print('Forest output is '+forest_target_names[int(prediction4[i])]+'\n')
```

```
["'Sugi' forest", "'Hinoki' forest", "'Mixed deciduous' forest", "'Other' non-forest land"]
record idx i is 38
For this Hinoki record,component tree 1 is classify as 'Hinoki' forest
For this Hinoki record,component tree 2 is classify as 'Sugi' forest
For this Hinoki record,component tree 3 is classify as 'Hinoki' forest
For this Hinoki record,component tree 4 is classify as 'Sugi' forest
Forest output is 'Sugi' forest

record idx i is 286
For this Hinoki record,component tree 1 is classify as 'Mixed deciduous' forest
For this Hinoki record,component tree 2 is classify as 'Sugi' forest
For this Hinoki record,component tree 3 is classify as 'Hinoki' forest
For this Hinoki record,component tree 4 is classify as 'Sugi' forest
Forest output is 'Sugi' forest

record idx i is 290
For this Hinoki record,component tree 1 is classify as 'Sugi' forest
For this Hinoki record,component tree 2 is classify as 'Sugi' forest
For this Hinoki record,component tree 3 is classify as 'Hinoki' forest
For this Hinoki record,component tree 4 is classify as 'Hinoki' forest
Forest output is 'Sugi' forest

record idx i is 313
For this Hinoki record,component tree 1 is classify as 'Sugi' forest
For this Hinoki record,component tree 2 is classify as 'Hinoki' forest
For this Hinoki record,component tree 3 is classify as 'Hinoki' forest
For this Hinoki record,component tree 4 is classify as 'Sugi' forest
Forest output is 'Sugi' forest

record idx i is 324
For this Hinoki record,component tree 1 is classify as 'Sugi' forest
For this Hinoki record,component tree 2 is classify as 'Sugi' forest
For this Hinoki record,component tree 3 is classify as 'Hinoki' forest
For this Hinoki record,component tree 4 is classify as 'Hinoki' forest
Forest output is 'Sugi' forest
```

We can see that most of these Hinoki records are classified as “Sugi” in the component tree, which makes the final output of the forest becoming “Sugi” instead of “Hinoki”.

To look further, we try to trace the transversal path to see the underlying reason behind the misclassification. We would take the record with index 286 as an example.

```
record idx i is 286
For this Hinoki record,component tree 1 is classify as 'Mixed deciduous' forest
[[1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
For this Hinoki record,component tree 2 is classify as 'Sugi' forest
[[1 1 0 0 0 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
For this Hinoki record,component tree 3 is classify as 'Hinoki' forest
[[1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0]]
For this Hinoki record,component tree 4 is classify as 'Sugi' forest
[[1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0]]
Forest output is 'Sugi' forest
```

The paths are highlighted on corresponding component trees:

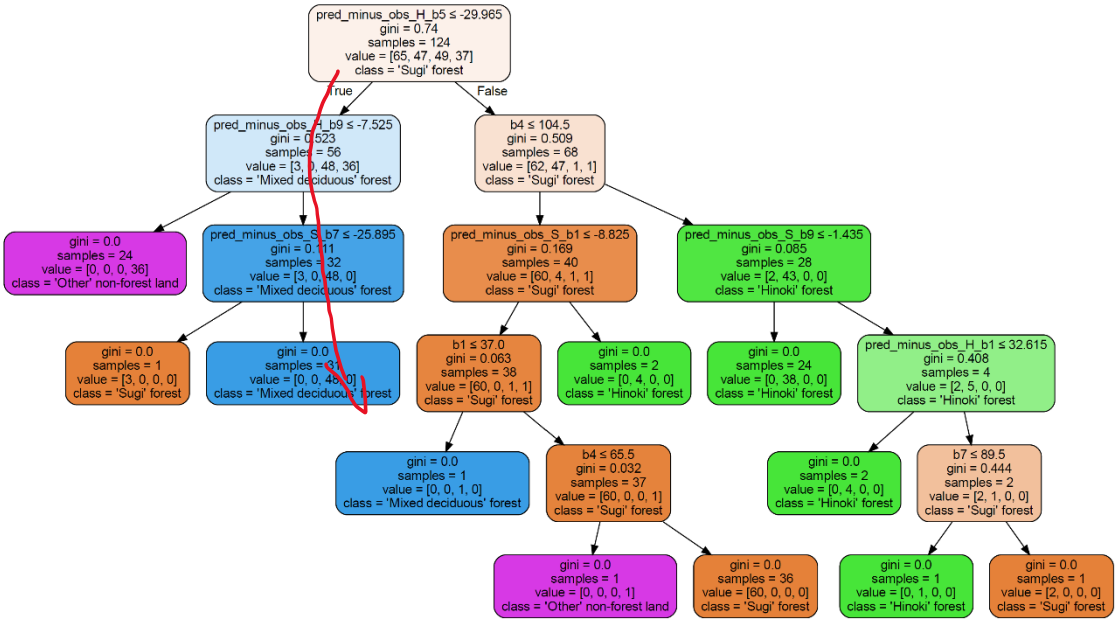


Figure 10 n_estimator=4, Tree1

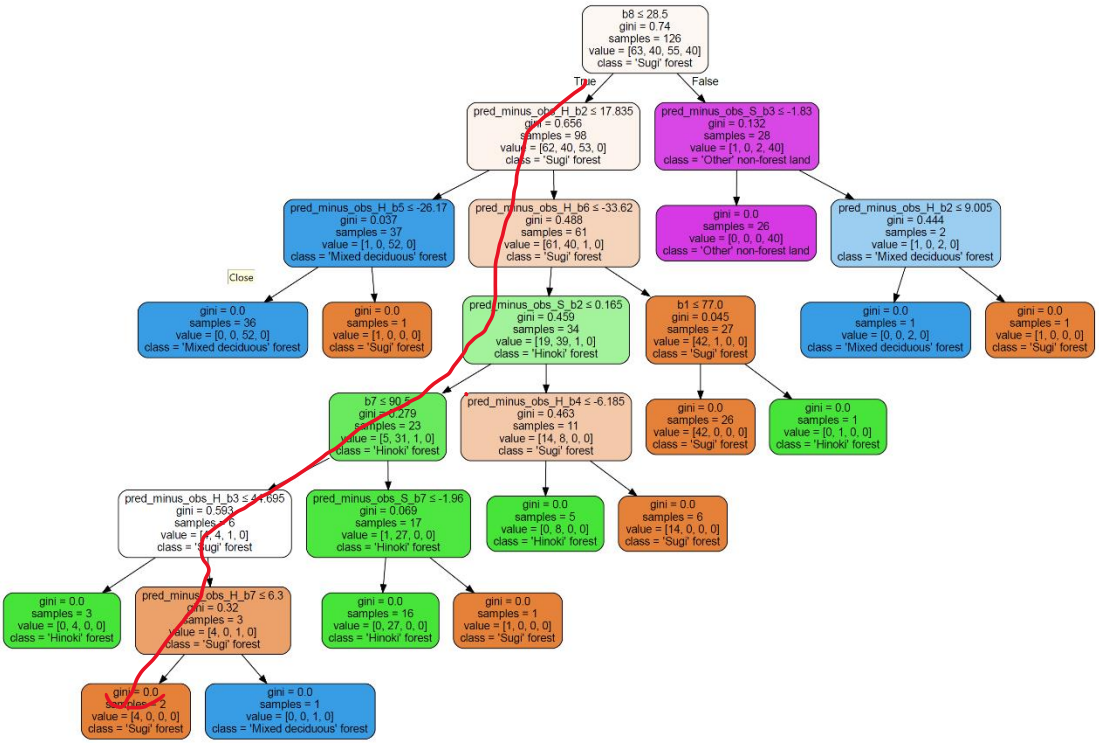


Figure 11 n_estimator=4, Tree2

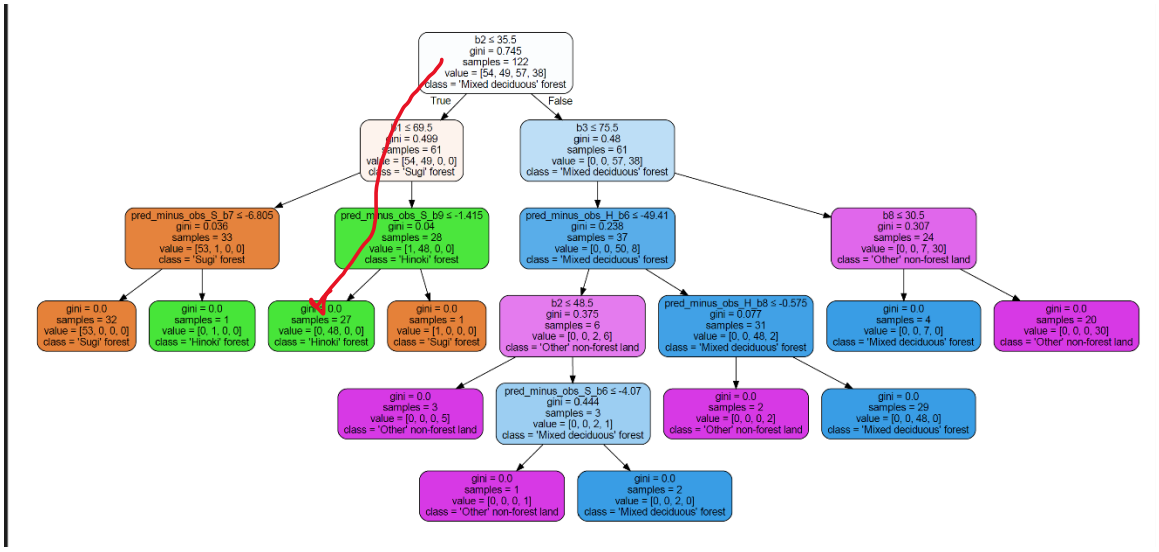


Figure 12 n_estimator=4, Tree3

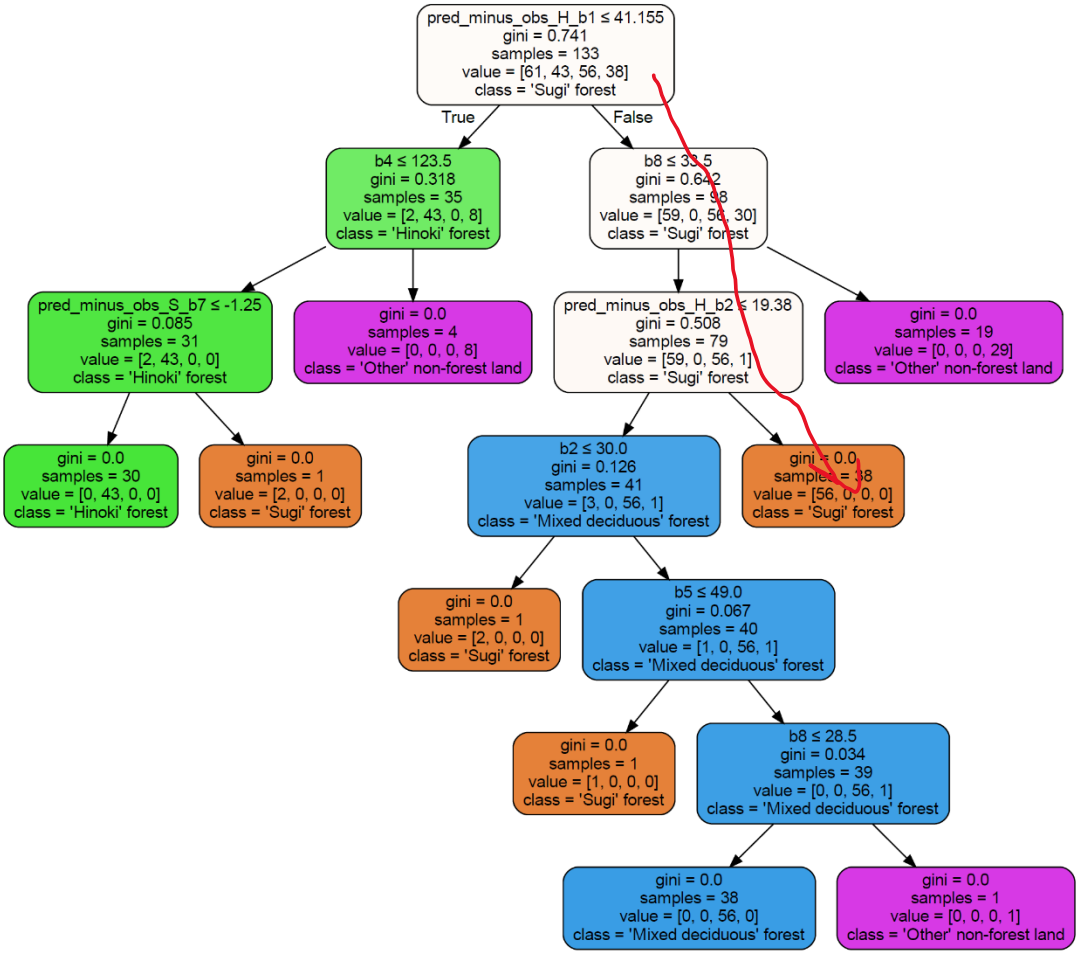


Figure 13 n_estimator=4, Tree4

We can see that different order of attributes and different values of threshold can give a very different output result in every component tree. This kind of asynchronization can give rise to the misclassification in the finalized output in the forest. In this case, it is “Hinoki being misclassified as Sugi”.

The case is similar for the samples with truth value “Mixed”. Though component tree 1 can obtain a high number of correct predictions (78), 7 of those records can get misclassified when they come to the random forest. Among the 7 records, 4 of them being misclassified as “Sugi” and 3 of them being misclassified as “Other”:

```
i]: import numpy as np
    x=prediction4_1.size
    forestTest= np.array(forest_data2)
    forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
    print(forest_target_names)
    for i in range(x):
        if prediction4_1[i]==2 and forest_target2[i]==2 and prediction4[i]!=2: #classify correct in tree 1 but not in forest
            print('record idx i is '+ str(i))
            print('For this Mixed record,component tree 1 is classify as '+forest_target_names[int(prediction4_1[i])])
            print(clf4_1.decision_path(forestTest).todense()[i])
            print('For this Mixed record,component tree 2 is classify as '+forest_target_names[int(prediction4_2[i])])
            print(clf4_2.decision_path(forestTest).todense()[i])
            print('For this Mixed record,component tree 3 is classify as '+forest_target_names[int(prediction4_3[i])])
            print(clf4_3.decision_path(forestTest).todense()[i])
            print('For this Mixed record,component tree 4 is classify as '+forest_target_names[int(prediction4_4[i])])
            print(clf4_4.decision_path(forestTest).todense()[i])
            print('Forest output is '+forest_target_names[int(prediction4[i])]+'\n')

    ["'Sugi' forest", "'Hinoki' forest", "'Mixed deciduous' forest", "'Other' non-forest land"]
    record idx i is 22
    For this Mixed record,component tree 1 is classify as 'Mixed deciduous' forest
    [[1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 2 is classify as 'Sugi' forest
    [[1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0]]
    For this Mixed record,component tree 3 is classify as 'Sugi' forest
    [[1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 4 is classify as 'Sugi' forest
    [[1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0]]
    Forest output is 'Sugi' forest

    record idx i is 112
    For this Mixed record,component tree 1 is classify as 'Mixed deciduous' forest
    [[1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 2 is classify as 'Other' non-forest land
    [[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0]]
    For this Mixed record,component tree 3 is classify as 'Other' non-forest land
    [[1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0]]
    For this Mixed record,component tree 4 is classify as 'Other' non-forest land
    [[1 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0]]
    Forest output is 'Other' non-forest land

    record idx i is 139
    For this Mixed record,component tree 1 is classify as 'Mixed deciduous' forest
    [[1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 2 is classify as 'Other' non-forest land
    [[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0]]
    For this Mixed record,component tree 3 is classify as 'Sugi' forest
    [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 4 is classify as 'Other' non-forest land
    [[1 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0]]
    Forest output is 'Other' non-forest land

    record idx i is 198
    For this Mixed record,component tree 1 is classify as 'Mixed deciduous' forest
    [[1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 2 is classify as 'Sugi' forest
    [[1 1 0 0 0 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 3 is classify as 'Sugi' forest
    [[1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 4 is classify as 'Sugi' forest
    [[1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0]]
    Forest output is 'Sugi' forest

    record idx i is 230
    For this Mixed record,component tree 1 is classify as 'Mixed deciduous' forest
    [[1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 2 is classify as 'Sugi' forest
    [[1 1 0 0 0 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 3 is classify as 'Sugi' forest
    [[1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 4 is classify as 'Sugi' forest
    [[1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0]]
    Forest output is 'Sugi' forest

    record idx i is 234
    For this Mixed record,component tree 1 is classify as 'Mixed deciduous' forest
    [[1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 2 is classify as 'Sugi' forest
    [[1 1 0 0 0 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 3 is classify as 'Sugi' forest
    [[1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 4 is classify as 'Sugi' forest
    [[1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0]]
    Forest output is 'Sugi' forest

    record idx i is 270
    For this Mixed record,component tree 1 is classify as 'Mixed deciduous' forest
    [[1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
    For this Mixed record,component tree 2 is classify as 'Other' non-forest land
    [[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0]]
    For this Mixed record,component tree 3 is classify as 'Other' non-forest land
    [[1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1]]
    For this Mixed record,component tree 4 is classify as 'Other' non-forest land
    [[1 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0]]
    Forest output is 'Other' non-forest land
```

The main reason behind is due to the misclassification in other component trees. Therefore, even though tree 1 can output more correct labels for “Mixed”, it is shown that we still need other trees to achieve good performance for the random forest to perform well. This also explains why the correct classification for mixed has dropped in random forest compared to using component tree 1 solely.

(c)For a random forest classifier (or one of its component trees), the relative importance of the attributes can be measured through the field of the classifier. For selected component trees in (b), compare their associated lists of relative attribute importance values. (25%)

```
print(clf4_1.feature_importances_ )
print(clf4_2.feature_importances_ )
print(clf4_3.feature_importances_ )
print(clf4_4.feature_importances_ )
```

```
[0.01321829 0.         0.         0.29641025 0.         0.
 0.00910345 0.         0.         0.01040394 0.         0.
 0.         0.30380679 0.         0.         0.         0.27221711
 0.04969522 0.         0.         0.         0.         0.
 0.03855578 0.         0.00658916]
[0.01333301 0.         0.         0.         0.         0.
 0.02090173 0.26677032 0.         0.         0.35015067 0.02548087
 0.06949328 0.01339291 0.14186448 0.01092037 0.         0.
 0.         0.0449005  0.02962892 0.         0.         0.
 0.01316295 0.         0.        ]
[0.32186017 0.35357365 0.13873392 0.         0.         0.
 0.         0.07698731 0.         0.         0.         0.
 0.         0.         0.04715745 0.         0.02604371 0.
 0.         0.         0.         0.         0.         0.00904295
 0.01331324 0.         0.01328761]
[0.         0.02501145 0.         0.08864794 0.01316392 0.
 0.         0.24653063 0.         0.25043766 0.35016034 0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.02604805 0.         0.        ]
```

The importance of every features in the 4 component trees are listed in tables below:

	b1	b2	b3	b4	b5	b6	b7	b8	b9	pred_minus_obs_H_b1	pred_minus_obs_H_b2	pred_minus_obs_H_b3	pred_minus_obs_H_b4	pred_minus_obs_H_b5	pred_minus_obs_H_b6	pred_minus_obs_H_b7	pred_minus_obs_H_b8	pred_minus_obs_H_b9	pred_minus_obs_S_b1	pred_minus_obs_S_b2	pred_minus_obs_S_b3	pred_minus_obs_S_b4	pred_minus_obs_S_b5	pred_minus_obs_S_b6	pred_minus_obs_S_b7	pred_minus_obs_S_b8	pred_minus_obs_S_b9
Tree 1	0.013218	0	0	0.296410	0	0	0.009103	0	0.006589	0	0.010404	0	0	0	0.301807	0	0	0	0.272217	0.049695	0	0	0	0	0.038558	0	0.006589
Tree 2	0.013333	0	0	0	0	0	0.026692	0.26677	0	0	0	0.350151	0.025481	0.069493	0.013393	0.141864	0.010920	0	0	0	0.044901	0.029629	0	0	0	0.013163	0	0
Tree 3	0.32186	0.353574	0.138734	0	0	0	0.076967	0	0	0	0	0	0	0	0.047157	0	0.026044	0	0	0	0	0	0	0	0.009043	0.013313	0	0.013288
Tree 4	0	0.025011	0	0.088648	0.013164	0	0	0.248531	0	0.250438	0.350160	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.026048	0	0

For every attribute, the highest importance is highlighted as green and the lowest importance value is highlighted as red. In this way, the supplementary relationship among the trees can be visualized through the table.

We can see that every trees are utilizing different important attributes. For example, attributes b4, pred_minus_obs_H_b5, pred_minus_obs_H_b9, pred_minus_obs_S_b1, pred_minus_obs_S_b7 (Total: 5 attributes) are important for tree 1 compared to other trees.

B7, b8, pred_minus_obs_H_b3, pred_minus_obs_H_b4, pred_minus_obs_H_b6, pred_minus_obs_H_b7, pred_minus_obs_S_b2, pred_minus_obs_S_b3 (Total: 8 attributes) are important for tree 2.

B1,B2,B3, pred_minus_obs_H_b8, pred_minus_obs_S_b6, pred_minus_obs_S_b9 (Total: 6 attributes) are important for tree 3.

B5, pred_minus_obs_H_b1, pred_minus_obs_H_b2 (Total: 3 attributes) are important for tree 4.

It can be said that the number of important attributes for each of these component trees are distributed evenly. They can be supplementary to each other. For example, b1 is not taken as very important for tree 1, 2, and 4 (value around 0.01). Meanwhile, this attribute is taken seriously (value=0.32) for tree 3. For another example, pred_minus_obs_H_b9 is taken importantly in tree 1 (value=0.27), but it has never been used for constructing tree2, 3, and 4.

(d)Construct a naïve Bayes classifier model based on our data set, and compare the classification performance with that of the random forest model. (25%)

RANDOM FOREST

```
[120]: ##RANDOM FOREST ACCURACY
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
print('when n_estimator=6')
print(accuracy_score(forest_target2,prediction6))
print(confusion_matrix(forest_target2,prediction6))
print('\n\n')
print('when n_estimator=5')
print(accuracy_score(forest_target2,prediction5))
print(confusion_matrix(forest_target2,prediction5))
print('\n\n')
print('when n_estimator=4')
print(accuracy_score(forest_target2,prediction4))
print(confusion_matrix(forest_target2,prediction4))
print('\n\n')
print('when n_estimator=3')
print(accuracy_score(forest_target2,prediction3))
print(confusion_matrix(forest_target2,prediction3))
print('\n\n')
print('when n_estimator=2')
print(accuracy_score(forest_target2,prediction2))
print(confusion_matrix(forest_target2,prediction2))
print('\n\n')
```

```
when n_estimator=6
0.803076923076923
[[123   6   7   0]
 [ 11  27   0   0]
 [ 21   0  75   9]
 [  3   0   7  36]]


when n_estimator=5
0.7876923076923077
[[113  12  11   0]
 [  5  32   1   0]
 [ 14   3  78  10]
 [  3   0  10  33]]


when n_estimator=4
0.8246153846153846
[[127   6   3   0]
 [ 10  28   0   0]
 [ 16   0  76  13]
 [  1   0   8  37]]


when n_estimator=3
0.7569230769230769
[[115  15   6   0]
 [ 11  27   0   0]
 [ 21   0  71  13]
 [  3   1   9  33]]


when n_estimator=2
0.7476923076923077
[[121  13   2   0]
 [ 10  28   0   0]
 [ 26   7  63   9]
 [  1   3  11  31]]
```

NAÏVE BAYES

```
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
nb=nb.fit(forest_data,forest_target)
prediction=nb.predict(forest_data2)

print('\n')
print(accuracy_score(forest_target2,prediction))
print(confusion_matrix(forest_target2,prediction))
```

```
0.803076923076923
[[113  12  11   0]
 [  8  30   0   0]
 [ 13   0  81  11]
 [  1   0   8  37]]
```

First, we compare the classification performance in terms of accuracy score. We can see that the accuracy of Naïve Bayes classifier is the same as the accuracy score achieved by random forest with n_estimator=6. Meanwhile, random forest with n_estimator=6 is the second best random forest we have constructed. Therefore, we can conclude that Naïve Bayes classifier is performing well, it is better than the 4 forests we have constructed. The accuracy score is only worse when compared to random forest with n_estimator=4.

Compared to random forest with n_estimator=6, Naived Bayes classifier misclassify 10 more samples in “Sugi”, 3 less samples in “Hinoki”,6 less samples in “Mixed” and 1 less samples in “Other”. We can say Naïve Bayes is not doing so well in classifying “Sugi”, but still do a better job in classifying other categories. Overall, the classification performance can be said to be very similar.

Compared to random forest with n_estimator=5, Naived Bayes classifier achieve same number of misclassified samples in classifying “Sugi” (Both get 23 misclassified samples). They are both not good at classifying “Sugi”. For “Hinoki”, Naïve Bayes classifier misclassified two more samples compared to random forest. For “Mixed”, Naïve Bayes classifier misclassified three less samples compared to random forest. For “Other”, Naïve Bayes classifier misclassified four less samples compared to random forest. Again, the overall result can be said to be similar, just that Naïve Bayes is slightly doing better in classifying “Other” and “Mixed”, slightly worse in classifying “ Hinoki”.

Compared to random forest with n_estimator=4, Naïve Bayes classifier misclassified 14 more samples in classifying “Sugi”. It is performing slightly better when classifying “Hinoki” and “Mixed” (with 2 and 5 less misclassified samples respectively). The performance in classifying “Other” is just the same. We can say that Naïve Bayes is not doing so well in classifying “Sugi”, giving a lower accuracy score compared to those of the random forest.

Compared to random forest with n_estimator=3, Naïve Bayes classifier misclassified 2 more samples in classifying “Sugi”, 3 less samples in classifying “Hinoki”, 10 less samples in classifying “Mixed” and 4 less samples in classifying “Other”. Apart from the slightly worse performance in classifying “Sugi”, we can say that Naïve Bayes classifier is performing better compared to random forest with n_estimator=3, especially in classifying “Mixed”.

Compared to random forest with n_estimator=2, Naïve Bayes classifier is performing worse in classifying “Sugi” (with 8 more misclassified samples), but it is achieving better classification result when classifying the others (2 less misclassified samples for “Hinoki” ,18less misclassified samples for “Mixed”, 6 less misclassified samples for “Other”). We can say that the Naïve Bayes is not doing so well in classifying “Sugi” in this case, but classifying “Mixed” so much better compared to the random forest.

To have a clear understanding on the comparison mentioned above, the results are summarized in table below:

	Misclassified samples in Forest (n_est=6)	Misclassified samples in Naïve Bayes	😊 refers to improvement 😟 refers to poorer performance The numbers is the difference of misclassified samples
Sugi	13	23	😟 10
Hinoki	11	8	😊 3
Mixed	30	24	😊 6
Other	10	9	😊 1

	Misclassified samples in Forest (n_est=5)	Misclassified samples in Naïve Bayes	😊 refers to improvement 😟 refers to poorer performance The numbers is the difference of misclassified samples
Sugi	23	23	-
Hinoki	6	8	😟 2
Mixed	27	24	😊 3
Other	13	9	😊 4

	Misclassified samples in Forest (n_est=4)	Misclassified samples in Naïve Bayes	😊 refers to improvement 😟 refers to poorer performance The numbers is the difference of misclassified samples
Sugi	9	23	😟 14
Hinoki	10	8	😊 2
Mixed	29	24	😊 5
Other	9	9	-

	Misclassified samples in Forest (n_est=3)	Misclassified samples in Naïve Bayes	😊 refers to improvement 😟 refers to poorer performance The numbers is the difference of misclassified samples
Sugi	21	23	😟 2
Hinoki	11	8	😊 3
Mixed	34	24	😊 10
Other	13	9	😊 4

	Misclassified samples in Forest (n_est=2)	Misclassified samples in Naïve Bayes	😊 refers to improvement 😟 refers to poorer performance The numbers is the difference of misclassified samples
Sugi	15	23	😟8
Hinoki	10	8	😊2
Mixed	42	24	😊18
Other	15	9	😊6

By look at all the tables, we can summarize that :

For classifying "Sugi", Naïve Bayes classifier is performing **poor** to every random forest.

For classifying "Hinoki", Naïve Bayes classifier is performing slightly better compared to most (4 out of 5) random forest. Their performance can be said to be very similar.

For classifying "Mixed", Naïve Bayes classifier is performing **better** compared to every random forest.

For classifying "Other", Naïve Bayes classifier is performing slightly better compared to all random forest.