香港城市大學
City University of Hong Kong

CS3481 Fundamental of Data Science

Assignment 1 Report

SID: 56228799

Name: Chan Tsz Fung

Contents

(a) Construct multiple decision trees based on the default training set/test set partition using different parameter settings. Compare the structures and classification performances of these decision trees. (25%)

We have constructed a total of 6 decision trees using different parameters in criterion, min_samples_split and max_leaf_nodes.

First, we try out using entropy criterion and max depth=3:

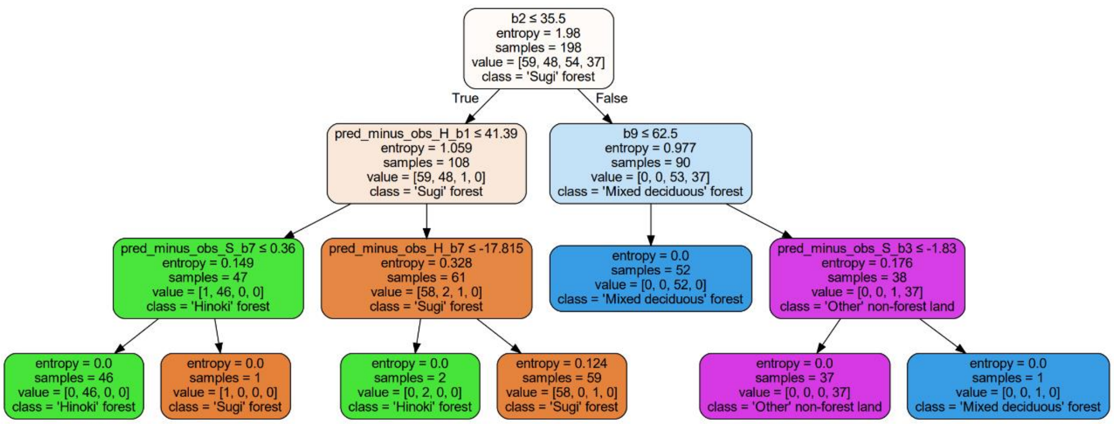


Figure 1 Tree1

Then, we changed the criterion from entropy to gini index, while keeping the max depth remains the same:

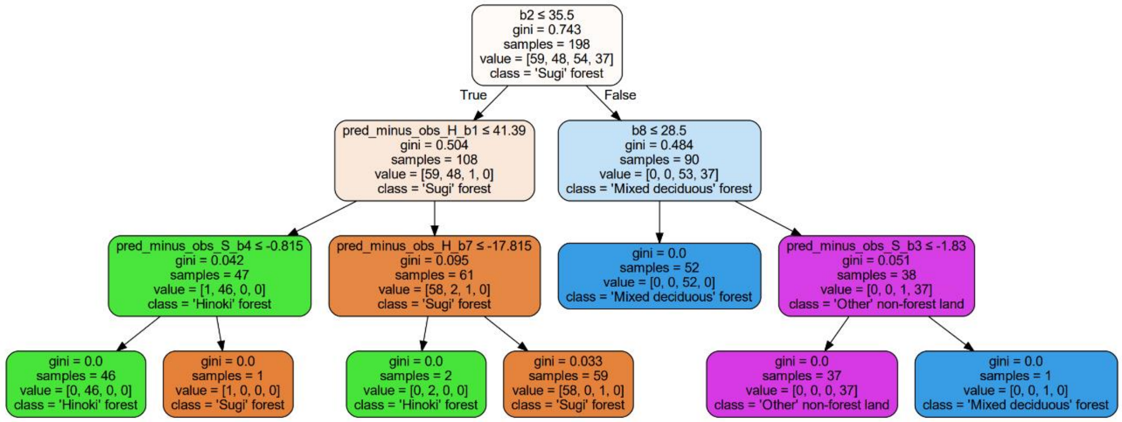


Figure 2 Tree 2

<u>Structure comparison:</u>

We can see that the size of both trees are the same and having the same number of leaf nodes (7). Most of the attributes used in intermittent nodes are the same, except that node 3 and node 9 (in DFS order) use a different parameter and a different threshold. In addition, every corresponding node values in the two trees above are the same. The transversing path for all the classes is about 3.

Classification performance comparison:

```
In [80]: #build decision tree 1
         from sklearn import tree
         clf=tree.DecisionTreeClassifier(max_depth=3)
         clf=clf.fit(forest_data,forest_target)

         import numpy as np
         forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
         forestTest= np.array(forest_data2)
         prediction= clf.predict(forestTest)
         # print(forest_target_names[prediction])

         from sklearn.metrics import accuracy_score
         print(accuracy_score(forest_target2,prediction))

         from sklearn.metrics import confusion_matrix
         print(confusion_matrix(forest_target2,prediction))

         0.7815384615384615
         [[119  12   5   0]
          [  9  29   0   0]
          [ 20   0  75  10]
          [  1   1  13  31]]
```

```
In [81]: #build decision tree 2

         from sklearn import tree
         clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy")
         clf=clf.fit(forest_data,forest_target)

         import numpy as np
         forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
         forestTest= np.array(forest_data2)
         prediction= clf.predict(forestTest)
         # print(forest_target_names[prediction])

         from sklearn.metrics import accuracy_score
         print(accuracy_score(forest_target2,prediction))

         from sklearn.metrics import confusion_matrix
         print(confusion_matrix(forest_target2,prediction))

         0.7815384615384615
         [[119  12   5   0]
          [  9  29   0   0]
          [ 20   0  74  11]
          [  1   1  12  32]]
```

Both classifier achieve the same classification performance with an accuracy rate of 78.15%. Both can be said to have good classification performance.

After constructing tree 1 and tree 2, we have discovered that overfitting occurs in the constructed two trees. (i.e. there are too little samples in some leaf nodes). Therefore, we try to make min_sample_split=50 such that only nodes with more than 50 samples would be further split. We have constructed the trees by adding the parameter in both entropy and gini index case:

The result trees 3 and 4 are as follows:

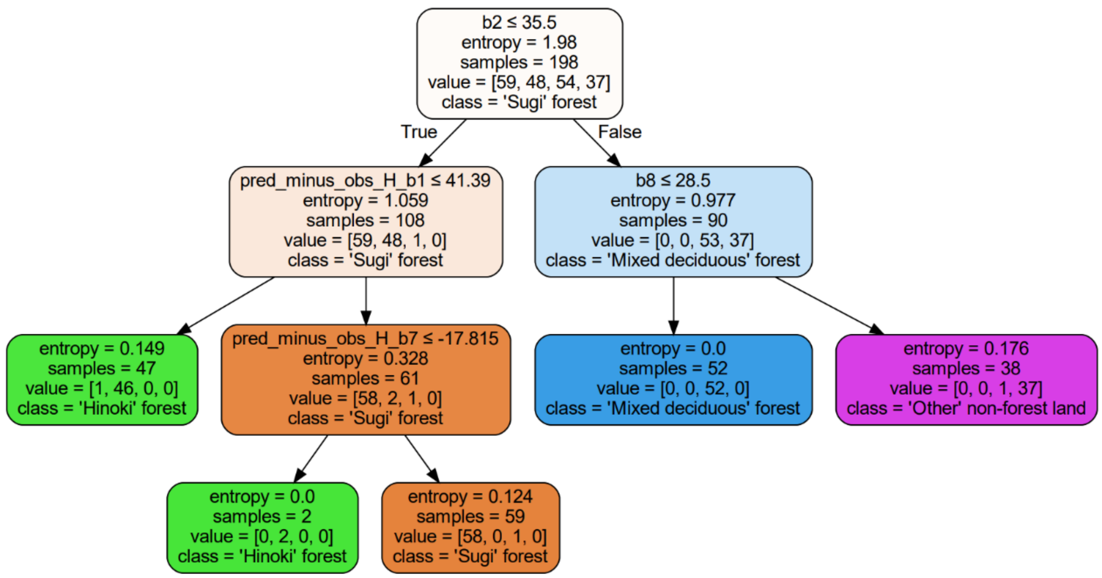


Figure 3 Tree 3

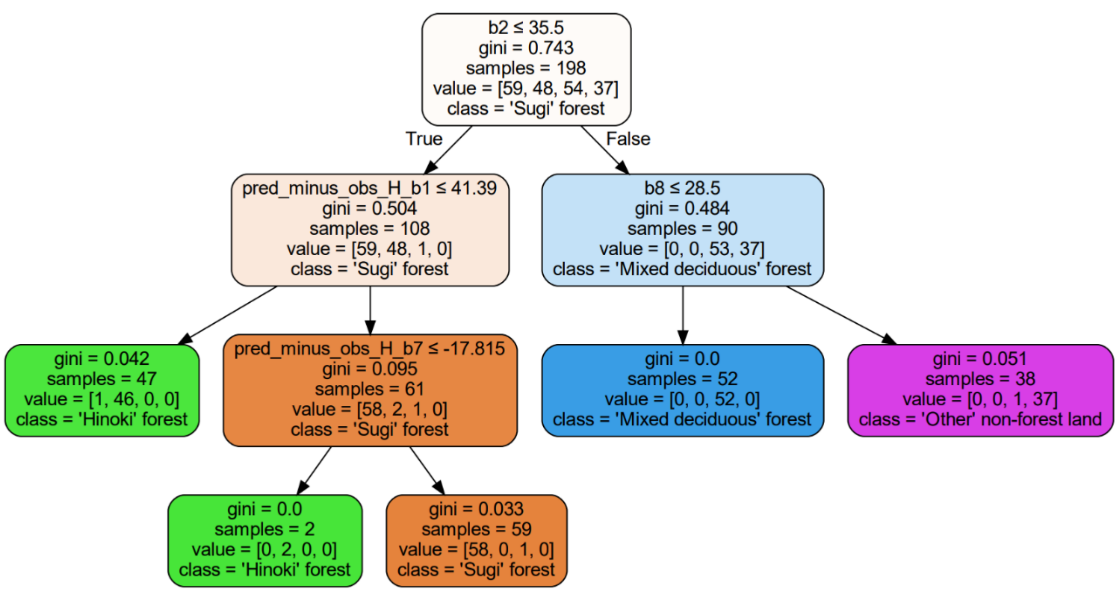


Figure 4 Tree 4

Structure comparison:

Again, the constructed two trees have very similar structure, they are of the same size and have same number of leaf nodes. Compared to the previously constructed tree 1 and tree 2, it is obvious that the number of leaf nodes is reduced from 7 to 5. Also, the path for classifying "Hinoki" forest (majority) and the path for classifying "Other" non-forest land have been reduced to two.

Classification performance comparison:

```
In [82]: #build decision tree 3
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy",min_samples_split=50)
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))

0.7876923076923077
[[119  12   5   0]
 [  9  29   0   0]
 [ 20   0  75  10]
 [  1   1  11  33]]
```

```
In [84]: #build decision tree 4
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,min_samples_split=50)
clf=clf.fit(forest_data,forest_target)


import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))

0.7876923076923077
[[119  12   5   0]
 [  9  29   0   0]
 [ 20   0  74  11]
 [  1   1  10  34]]
```

After cutting out leaf nodes using min_sample_split, we can see that both the gini classifier and the entropy classifier improve in their accuracy. The accuracy rate of both classifiers are improved from 0.7815 to 0.7877 compared as the previously constructed ones.

After constructing tree 3 and tree 4, we can observe that overfitting still occurs. (There is one node with just 2 samples). The last strategy we adopt is adding parameter max_leaf_node=4. This would enhance the generalization of our model.

The modified trees (gini & entropy) are displayed as follows:

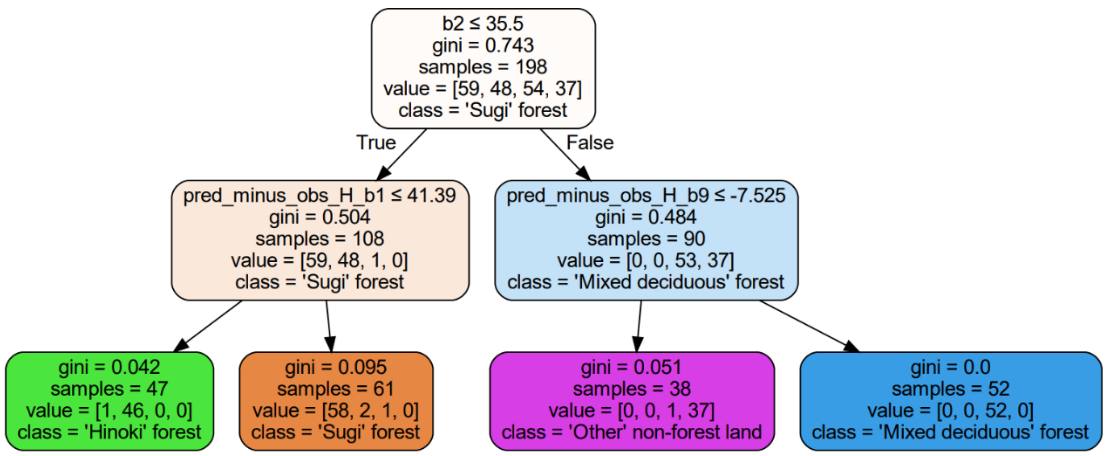


Figure 5 Tree 5

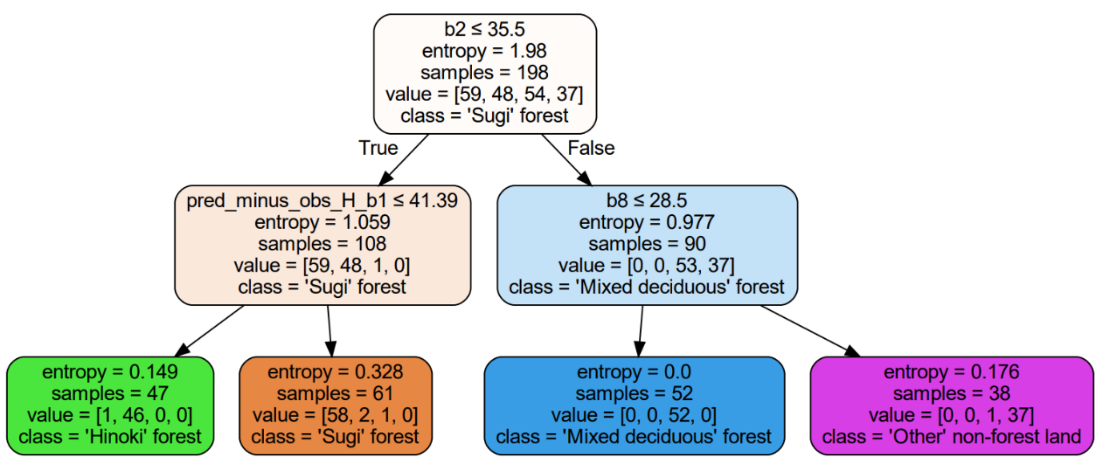


Figure 6 Tree6

<u>Structure comparison:</u>

We can see that the newly constructed two trees are having smaller size compared to all the other trees constructed previously. The number of leaf nodes is reduced to 4. The path length for all different classes is just 2 now.

Classification performance:

```
In [85]: #build decision tree5
         from sklearn import tree
         clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy",min_samples_split=50,max_leaf_nodes=4)
         clf=clf.fit(forest_data,forest_target)
         import numpy as np
         forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
         forestTest= np.array(forest_data2)
         prediction= clf.predict(forestTest)
         # print(forest_target_names[prediction])

         from sklearn.metrics import accuracy_score
         print(accuracy_score(forest_target2,prediction))

         from sklearn.metrics import confusion_matrix
         print(confusion_matrix(forest_target2,prediction))

         0.8
         [[126   5   5   0]
          [ 12  26   0   0]
          [ 20   0  75  10]
          [  2   0  11  33]]
```

```
In [86]: #build decision tree6
         from sklearn import tree
         clf=tree.DecisionTreeClassifier(max_depth=3,min_samples_split=50,max_leaf_nodes=4)
         clf=clf.fit(forest_data,forest_target)
         import numpy as np
         forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
         forestTest= np.array(forest_data2)
         prediction= clf.predict(forestTest)
         # print(forest_target_names[prediction])

         from sklearn.metrics import accuracy_score
         print(accuracy_score(forest_target2,prediction))

         from sklearn.metrics import confusion_matrix
         print(confusion_matrix(forest_target2,prediction))

         0.8
         [[126   5   5   0]
          [ 12  26   0   0]
          [ 20   0  75  10]
          [  2   0  11  33]]
```

We can see that after restricting the number of leaf node to 4, both the entropy case and the gini index case achieved an improvement in classification performance. Both classifiers achieve an accuracy rate of 0.8. This is higher than 0.7815 and 0.7877 results from previously constructed classifier. This means that the final constructed tree classifier with fewer layers can be said to have better generalization performance.

(b) Exchange the training and test set and repeat the tasks in (a). (25%)

This time, we make use of forest_testing.csv as the training data, and make use of forest_training.csv as the testing data. Same as before, we build a total of 6 trees, by using the parameters like criterion, max_depth, min_sample_split, and maxleafnode.

First, we use max depth 3 as a parameter, construct the trees using both entropy values and gini index.

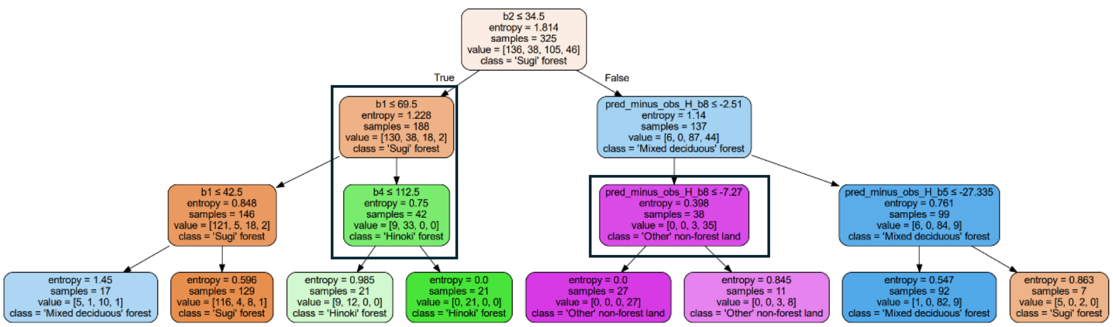


Figure 7 Tree 1(Test.csv)

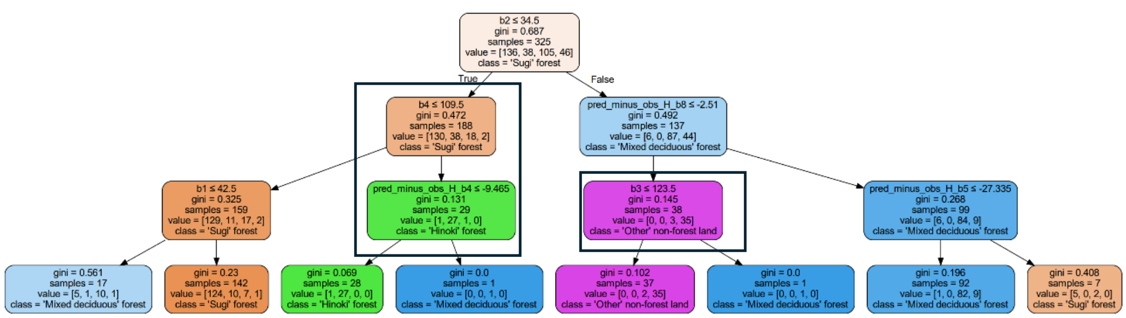


Figure 8Tree 2(Test.csv)

<u>Structure comparison</u>

We can see that both trees are having same size, with the same number of leaf node(8). The path length for transversing all the classes are 3. Regarding the attributes used, b1 is used as the second attribute in the left branch for entropy, while b4 is used as the second attribute in the left branch for gini index. On the other hand, pred_minus_obs_H_b8 is used as the third attribute in the right branch for entropy, while b3 is used as the third attribute in the right branch for gini index. Also, b4 us used as third attribute in the left branch of entropy case, while pred_minus_obs_H_b4 is the third attribute used in the left branch of gini index case The attributes used in all other nodes are just the same.

Classification performance

In [111]:
```
#build decision tree 1
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3)
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.8686868686868687
[[55  2  2  0]
 [12 32  4  0]
 [ 0  0 53  1]
 [ 0  0  5 32]]
```

In [110]:
```
#build decision tree 2

from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy")
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.9242424242424242
[[55  2  2  0]
 [ 6 41  1  0]
 [ 0  0 53  1]
 [ 0  0  3 34]]
```

Both classifiers works well in classifying the training data set. For the gini index one, it achieve an accuracy rate of 86.87 % while the tree constructed by entropy value achieve an accuracy rate of 92.42%.

Similar as part(a), we notice that there are case of overfitting in the trees constructed. (ie. There are nodes with only few samples.) Therefore, our next two trees being constructed would be adding the new parameter min_samples_split=50 such that nodes would only be further split if it has a minimum of 50 samples.

The result trees are as follows:

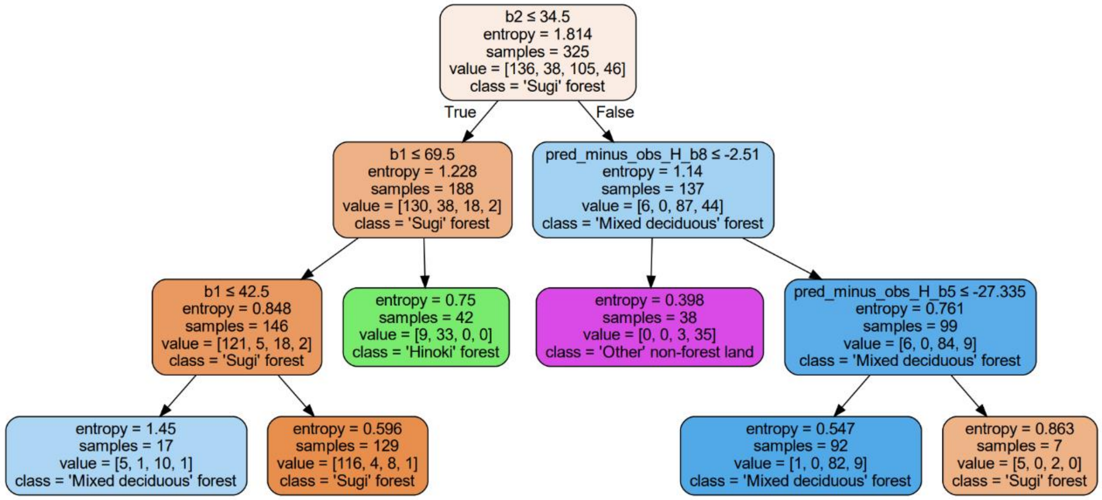


Figure 9 Tree 3(Test.csv)

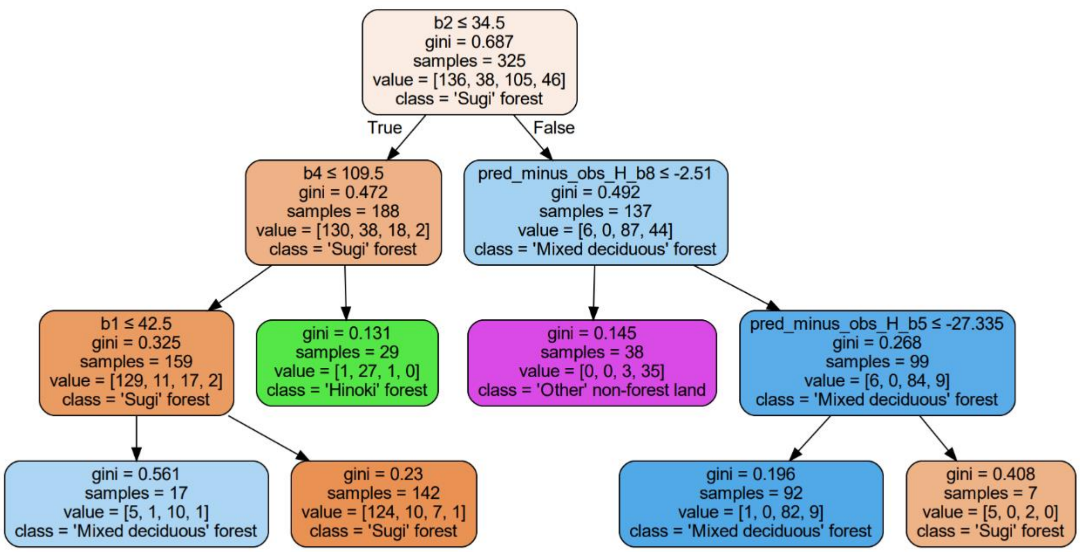


Figure 10 Tree 4(Test.csv)

Structure comparison

We can see that the size of the tree is getting smaller compared to the tree constructed before. The number of leaf node is reduced from 8 to 6. The path length for classifying into “Hinoki” and “Other” is shortened by 1.

Classification performance

```
In [112]: #build decision tree 3
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy",min_samples_split=50)
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))

0.9242424242424242
[[55  2  2  0]
 [ 6 41  1  0]
 [ 0  0 53  1]
 [ 0  0  3 34]]
```

```
In [113]: #build decision tree 4
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,min_samples_split=50)
clf=clf.fit(forest_data,forest_target)


import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))

0.8939393939393939
[[55  2  2  0]
 [12 35  1  0]
 [ 0  0 53  1]
 [ 0  0  3 34]]
```

Same as before, the tree constructed using gini index is having slightly lower accuracy rate compared to the tree constructed using entropy value (0.92>0.89). With the new parameter min_sample_split=50 used, the accuracy rate of the tree constructed by entropy value remains the same (0.9242) while the accuracy rate of the tree constructed by gini index achieved an improvement (from 0.8687 to 0.8939).

After using min_sample_split=50 in constructing new trees, we discovered that the tree being constructed is still having some nodes with only little samples. We would use the max_leaf_nodes=4 as the next parameter for constructing another 2 trees.

The results are as follows:

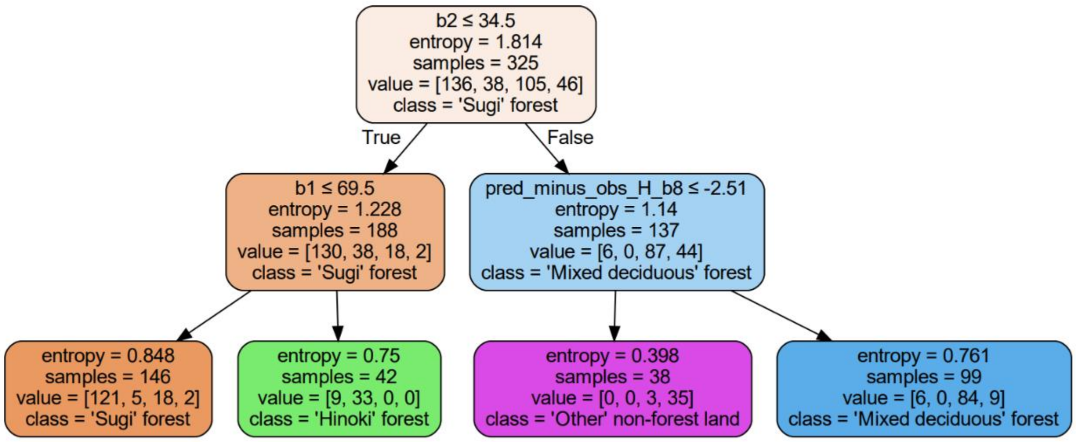


Figure 11 Tree 5(Test.csv)

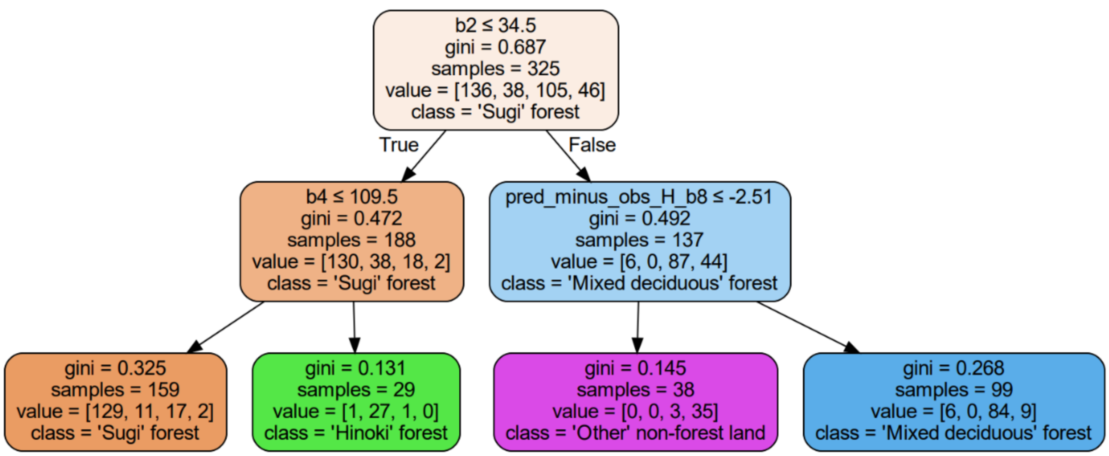


Figure 12 Tree 6 (Test.csv)

<u>Structure comparison</u>

We can see that by adding this new parameter, the size of the tree grows smaller. The number of leaf nodes has been reduced from 6 to 4, and the path for classifying all different classes is shorten to 2.

Classification performance

```
In [114]: #build decision tree5
          from sklearn import tree
          clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy",min_samples_split=50,max_leaf_nodes=4)
          clf=clf.fit(forest_data,forest_target)
          import numpy as np
          forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
          forestTest= np.array(forest_data2)
          prediction= clf.predict(forestTest)
          # print(forest_target_names[prediction])

          from sklearn.metrics import accuracy_score
          print(accuracy_score(forest_target2,prediction))

          from sklearn.metrics import confusion_matrix
          print(confusion_matrix(forest_target2,prediction))

          0.9242424242424242
          [[56  2  1  0]
           [ 6 41  1  0]
           [ 1  0 52  1]
           [ 0  0  3 34]]
```

```
In [115]: #build decision tree6
          from sklearn import tree
          clf=tree.DecisionTreeClassifier(max_depth=3,min_samples_split=50,max_leaf_nodes=4)
          clf=clf.fit(forest_data,forest_target)
          import numpy as np
          forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
          forestTest= np.array(forest_data2)
          prediction= clf.predict(forestTest)
          # print(forest_target_names[prediction])

          from sklearn.metrics import accuracy_score
          print(accuracy_score(forest_target2,prediction))

          from sklearn.metrics import confusion_matrix
          print(confusion_matrix(forest_target2,prediction))

          0.8939393939393939
          [[56  2  1  0]
           [12 35  1  0]
           [ 1  0 52  1]
           [ 0  0  3 34]]
```

Similar as before, the tree constructed using entropy achieve slightly higher accuracy rate compared to the one constructed using gini index (92%>89%). Both trees achieve high accuracy for the training set as test data, which means that the constructed tree can be considered as having good generalization performance.

(c) For selected trees in (a) and (b), observe the classification performance associated with the different classes, and determine which pair(s) of classes are likely to be confused with each other. (25%)

For part(a), the classification performance for different trees is shown below:

```
In [80]: #build decision tree 1
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3)
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))

0.7815384615384615
[[119  12   5   0]
 [  9  29   0   0]
 [ 20   0  75  10]
 [  1   1  13  31]]
```

```
In [81]: #build decision tree 2

from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy")
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))

0.7815384615384615
[[119  12   5   0]
 [  9  29   0   0]
 [ 20   0  74  11]
 [  1   1  12  32]]
```

In [82]:
```
#build decision tree 3
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy",min_samples_split=50)
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.7876923076923077
[[119  12   5   0]
 [  9  29   0   0]
 [ 20   0  75  10]
 [  1   1  11  33]]
```

In [84]:
```
#build decision tree 4
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,min_samples_split=50)
clf=clf.fit(forest_data,forest_target)


import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.7876923076923077
[[119  12   5   0]
 [  9  29   0   0]
 [ 20   0  74  11]
 [  1   1  10  34]]
```

In [85]:
```
#build decision tree5
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy",min_samples_split=50,max_leaf_nodes=4)
clf=clf.fit(forest_data,forest_target)
import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.8
[[126   5   5   0]
 [ 12  26   0   0]
 [ 20   0  75  10]
 [  2   0  11  33]]
```

In [86]:
```
#build decision tree6
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,min_samples_split=50,max_leaf_nodes=4)
clf=clf.fit(forest_data,forest_target)
import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.8
[[126   5   5   0]
 [ 12  26   0   0]
 [ 20   0  75  10]
 [  2   0  11  33]]
```

According to the non-diagonal value in the confusion matrix, we can conclude that "Mixed deciduous' forest" are likely to be wrongly classified as "Sugi' forest" (20 wrong classification for every confusion matrix constructed).

Apart from that, "Other non-forest land" is likely to be misclassified as "Mixed deciduous' forest" (more than 10 misclassified samples for every confusion matrix constructed)

The pair "Sugi" and "Hinoki" is also worth noticing, "Sugi' forest" is likely to be misclassified as "Hinoki' forest" (with 10 misclassified samples for first 4 confusion matrix constructed). In opposite direction, "Hinoki' forest" is likely to be misclassified as "Sugi' forest" (with about 10 misclassified samples for every confusion matrix constructed)

For part(b), the classification performance for different trees is shown below:

```
In [111]: #build decision tree 1
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3)
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))

0.8686868686868687
[[55  2  2  0]
 [12 32  4  0]
 [ 0  0 53  1]
 [ 0  0  5 32]]
```

```
In [110]: #build decision tree 2

from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy")
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))

0.9242424242424242
[[55  2  2  0]
 [ 6 41  1  0]
 [ 0  0 53  1]
 [ 0  0  3 34]]
```

In [112]:
```python
#build decision tree 3
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy",min_samples_split=50)
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.9242424242424242
[[55  2  2  0]
 [ 6 41  1  0]
 [ 0  0 53  1]
 [ 0  0  3 34]]
```

In [113]:
```python
#build decision tree 4
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,min_samples_split=50)
clf=clf.fit(forest_data,forest_target)


import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.8939393939393939
[[55  2  2  0]
 [12 35  1  0]
 [ 0  0 53  1]
 [ 0  0  3 34]]
```

In [114]:
```python
#build decision tree5
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,criterion="entropy",min_samples_split=50,max_leaf_nodes=4)
clf=clf.fit(forest_data,forest_target)
import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.9242424242424242
[[56  2  1  0]
 [ 6 41  1  0]
 [ 1  0 52  1]
 [ 0  0  3 34]]
```

In [115]:
```python
#build decision tree6
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3,min_samples_split=50,max_leaf_nodes=4)
clf=clf.fit(forest_data,forest_target)
import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.8939393939393939
[[56  2  1  0]
 [12 35  1  0]
 [ 1  0 52  1]
 [ 0  0  3 34]]
```

From the confusion matrix constructed, we can see that “Hinoki’ forest” is likely to be misclassified as “Sugi’ forest” (with about 6 misclassified samples for 3 confusion matrix constructed, 12 misclassified samples for another 3 confusion matrix constructed)

(d) For selected confused class pairs in (c), identify the corresponding leaf node(s) and analyze the sequence of decisions that lead to the misclassification. (25%)

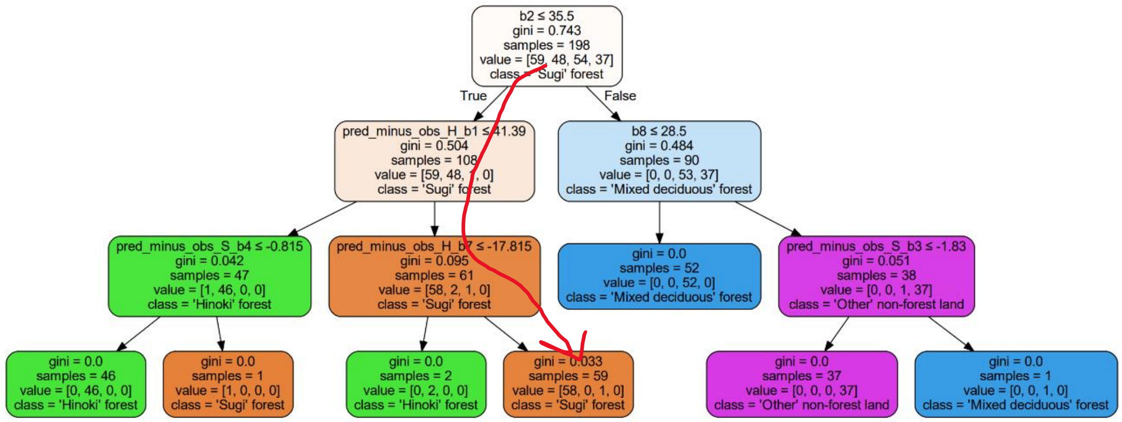
(d)(i). Misclassification in Tree built using training.csv

“Mixed deciduous’ forest” are likely to be wrongly classified as “Sugi’ forest”

```
In [140]: #build decision tree 1
          from sklearn import tree
          clf=tree.DecisionTreeClassifier(max_depth=3)
          clf=clf.fit(forest_data,forest_target)

          import numpy as np
          forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
          forestTest= np.array(forest_data2)
          prediction= clf.predict(forestTest)
          # print(forest_target_names[prediction])

          from sklearn.metrics import accuracy_score
          print(accuracy_score(forest_target2,prediction))

          from sklearn.metrics import confusion_matrix
          print(confusion_matrix(forest_target2,prediction))

          0.7846153846153846
          [[119  12   5   0]
           [  9  29   0   0]
           [ 20   0  73  12]
           [  1   1  10  34]]
```

```
In [148]: x=prediction.size
          for i in range(x):
              if prediction[i]==0 and forest_target2[i]==2:
                  print(clf.decision_path(forestTest).todense()[i])

          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
          [[1 1 0 0 0 1 0 1 0 0 0 0 0]]
```

We can see that for this kind of wrong classification, it transverses node 0,1,5,7, which means the highlighted path:



We then check the corresponding 20 records.

```
In [153]: x=prediction.size
          print(forest_feature_names2)
          for i in range(x):
              if prediction[i]==0 and forest_target2[i]==2:
                  print(i)
                  print(forest_data2[i])
```

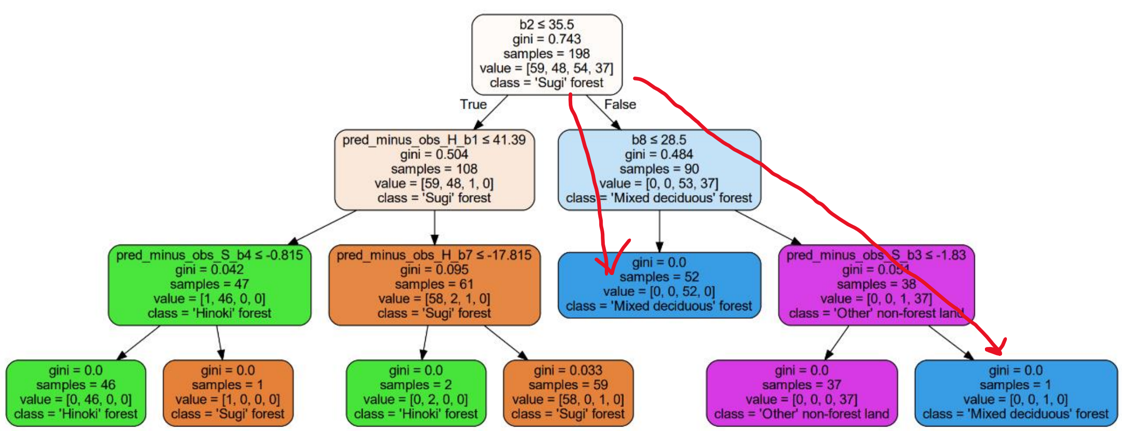
```
['b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8', 'b9', 'pred_minus_obs_H_b1', 'pred_minus_obs_H_b2', 'pred_minus_obs_H_b3', 'pr
ed_minus_obs_H_b4', 'pred_minus_obs_H_b5', 'pred_minus_obs_H_b6', 'pred_minus_obs_H_b7', 'pred_minus_obs_H_b8', 'pred_minus_obs
_H_b9', 'pred_minus_obs_S_b1', 'pred_minus_obs_S_b2', 'pred_minus_obs_S_b3', 'pred_minus_obs_S_b4', 'pred_minus_obs_S_b5', 'pre
d_minus_obs_S_b6', 'pred_minus_obs_S_b7', 'pred_minus_obs_S_b8', 'pred_minus_obs_S_b9']
17
[36.0, 35.0, 56.0, 90.0, 61.0, 105.0, 63.0, 23.0, 53.0, 82.41, 16.46, 42.18, 12.72, -34.79, -41.83, 17.88, 6.09, 2.32, -24.37,
-1.6, -4.84, -28.5, -1.14, -6.4, -24.61, -1.56, -4.98]
21
[50.0, 34.0, 56.0, 98.0, 55.0, 99.0, 84.0, 25.0, 55.0, 68.27, 17.41, 42.13, 4.33, -28.81, -35.9, -3.5, 4.04, 0.25, -24.69, -1.8
6, -5.51, -29.81, -1.33, -6.79, -25.22, -1.95, -5.12]
22
[31.0, 26.0, 47.0, 90.0, 52.0, 94.0, 53.0, 22.0, 48.0, 86.08, 25.75, 51.01, 13.74, -25.95, -31.4, 27.52, 8.09, 8.28, -24.26, -
1.74, -4.66, -28.43, -0.98, -5.94, -24.63, -2.51, -5.83]
26
[68.0, 29.0, 54.0, 112.0, 48.0, 94.0, 89.0, 25.0, 56.0, 48.57, 22.75, 43.9, -8.56, -22.01, -31.55, -8.86, 4.84, 0.05, -22.65, -
1.79, -4.73, -27.02, -0.99, -5.82, -23.26, -2.29, -5.41]
47
[42.0, 29.0, 50.0, 83.0, 54.0, 93.0, 70.0, 25.0, 54.0, 72.1, 23.03, 49.13, 17.72, -28.07, -30.82, 8.46, 5.58, 2.94, -20.4, -2.5
2, -7.88, -21.57, -1.22, -6.19, -21.21, -2.92, -6.13]
102
[50.0, 31.0, 52.0, 85.0, 48.0, 86.0, 73.0, 24.0, 51.0, 62.35, 18.93, 41.98, 12.27, -22.86, -26.41, 3.6, 6.39, 5.23, -19.6, -1.0
5, -5.24, -19.52, -1.06, -4.89, -17.84, -1.64, -4.2]
111
[42.0, 31.0, 53.0, 91.0, 56.0, 95.0, 76.0, 23.0, 53.0, 68.81, 18.72, 41.32, 3.72, -31.12, -36.33, -4.75, 6.08, 1.73, -15.81, 0.
07, -2.67, -19.74, -0.49, -2.65, -13.76, -1.61, -3.48]
114
[45.0, 30.0, 51.0, 96.0, 53.0, 96.0, 84.0, 23.0, 55.0, 72.13, 21.09, 45.25, 8.72, -27.51, -35.05, -2.89, 7.75, 1.88, -22.33, -
1.04, -5.18, -25.88, -0.88, -5.16, -22.03, -2.22, -4.94]
126
[35.0, 32.0, 53.0, 95.0, 55.0, 98.0, 70.0, 24.0, 51.0, 78.37, 18.87, 42.67, 4.82, -29.78, -38.17, 5.26, 6.04, 4.84, -12.68, -0.
97, -1.91, -17.8, -0.7, -3.07, -12.39, -1.95, -3.89]
130
[39.0, 35.0, 59.0, 93.0, 60.0, 101.0, 71.0, 23.0, 50.0, 74.11, 16.02, 38.38, 0.17, -34.6, -40.72, 2.23, 5.9, 4.8, -18.16, -0.7
8, -2.94, -18.92, -0.56, -3.1, -15.16, -0.95, -3.15]
139
[47.0, 33.0, 56.0, 90.0, 64.0, 102.0, 89.0, 29.0, 59.0, 67.22, 18.47, 39.66, 11.55, -38.61, -41.71, -11.48, 1.79, -2.35, -18.2
5, -1.04, -3.17, -22.6, -0.8, -4.18, -18.6, -2.38, -4.74]
146
[40.0, 29.0, 51.0, 92.0, 52.0, 96.0, 69.0, 23.0, 53.0, 73.32, 22.67, 44.69, 8.96, -26.53, -35.79, 7.71, 7.92, 3.55, -16.53, -1.
02, -2.51, -21.69, -0.88, -4.0, -17.53, -2.54, -4.69]
198
[34.0, 32.0, 51.0, 93.0, 61.0, 106.0, 81.0, 25.0, 55.0, 78.86, 19.83, 46.08, 5.19, -35.77, -46.24, -6.15, 5.34, 0.79, -18.53, -
1.25, -4.02, -21.63, -0.78, -4.07, -18.26, -2.42, -4.67]
214
[33.0, 31.0, 53.0, 106.0, 53.0, 104.0, 64.0, 23.0, 52.0, 85.35, 23.79, 50.51, -7.83, -27.45, -43.15, 14.01, 7.23, 4.05, -22.43,
-2.79, -7.63, -20.0, -1.07, -4.45, -19.11, -2.43, -4.98]
227
[66.0, 32.0, 55.0, 91.0, 49.0, 91.0, 82.0, 27.0, 58.0, 46.58, 15.75, 36.1, 2.47, -24.39, -33.09, -5.67, 2.1, -3.57, -22.52, -0.
82, -5.41, -19.85, -1.27, -4.4, -19.16, -1.02, -2.81]
230
[48.0, 33.0, 54.0, 107.0, 58.0, 104.0, 90.0, 25.0, 55.0, 70.16, 21.33, 48.76, -6.14, -32.49, -43.12, -11.04, 5.24, 1.05, -22.4
9, -2.81, -7.31, -22.78, -1.23, -5.12, -19.92, -2.19, -4.89]
234
[31.0, 29.0, 50.0, 88.0, 53.0, 95.0, 69.0, 23.0, 50.0, 81.0, 21.5, 44.9, 11.08, -27.97, -35.53, 6.26, 7.03, 5.4, -19.95, -0.59,
-3.93, -24.9, -1.1, -5.16, -20.47, -1.51, -3.99]
235
[39.0, 32.0, 51.0, 94.0, 54.0, 98.0, 76.0, 23.0, 51.0, 72.82, 18.37, 43.8, 4.66, -28.95, -38.5, -1.03, 6.72, 4.08, -19.93, -0.5
4, -3.87, -23.99, -1.07, -5.18, -20.14, -1.03, -3.58]
288
[40.0, 33.0, 58.0, 72.0, 56.0, 92.0, 63.0, 22.0, 50.0, 69.96, 15.72, 35.07, 27.19, -31.14, -33.07, 10.97, 7.45, 4.91, -18.57, -
0.31, -4.57, -21.21, -1.02, -4.84, -17.04, -0.88, -3.33]
304
[48.0, 33.0, 57.0, 93.0, 54.0, 95.0, 94.0, 25.0, 54.0, 65.83, 17.21, 37.9, 8.69, -29.21, -35.9, -17.21, 4.21, 0.69, -21.74, -1.
69, -5.25, -26.3, -1.22, -5.49, -20.12, -0.93, -3.52]
```

We can see that for all the records, b2<=35.5, which makes the classification fall into the left branch in the first layer. This is the key node contributing to the misclassification. If the records above are having b2>35.5, 19 out of 20 classifications would be correct in the second layer (ie. b8<=28.5). This analysis on misclassification can be applied to tree 1 to tree 6.

“Other non-forest land” is likely to be misclassified as “Mixed deciduous’ forest”

The transversal path for the misclassification is shown as below(node 0,8,9)and(node 0,8,10,12):

```
In [182]: #build decision tree 1
          from sklearn import tree
          clf=tree.DecisionTreeClassifier(max_depth=3)
          clf=clf.fit(forest_data,forest_target)

          import numpy as np
          forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
          forestTest= np.array(forest_data2)
          prediction= clf.predict(forestTest)
          # print(forest_target_names[prediction])

          from sklearn.metrics import accuracy_score
          print(accuracy_score(forest_target2,prediction))

          from sklearn.metrics import confusion_matrix
          print(confusion_matrix(forest_target2,prediction))

          0.7846153846153846
          [[119  12   5   0]
           [  9  29   0   0]
           [ 20   0  73  12]
           [  1   1  10  34]]

In [176]: x=prediction.size
          # print(forest_feature_names2)
          for i in range(x):
              if prediction[i]==2 and forest_target2[i]==3:
                  print(clf.decision_path(forestTest).todense()[i])
          #         print(i)
          #         print(forest_data2[i])

          [[1 0 0 0 0 0 0 0 1 1 0 0 0]]
          [[1 0 0 0 0 0 0 0 1 1 0 0 0]]
          [[1 0 0 0 0 0 0 0 1 1 0 0 0]]
          [[1 0 0 0 0 0 0 0 1 0 1 0 1]]
          [[1 0 0 0 0 0 0 0 1 1 0 0 0]]
          [[1 0 0 0 0 0 0 0 1 1 0 0 0]]
          [[1 0 0 0 0 0 0 0 1 1 0 0 0]]
          [[1 0 0 0 0 0 0 0 1 1 0 0 0]]
          [[1 0 0 0 0 0 0 0 1 0 1 0 1]]
          [[1 0 0 0 0 0 0 0 1 1 0 0 0]]
          [[1 0 0 0 0 0 0 0 1 1 0 0 0]]
          [[1 0 0 0 0 0 0 0 1 1 0 0 0]]
```



For the longer path, the reason for the misclassification is that the model fits too well (overfits) when being built. When the last node is pruned, the result should be correctly classified. (That is the improvement we have already made in tree 3 to 6)

For the shorter path, it is obvious that the key node for misclassification is b8<=28.5. When it is true, the record would be classified as "mixed" instead of "other", we can verify that by looking at all the misclassified records:

```
In [193]: x=prediction.size
          print(forest_feature_names2)
          for i in range(x):
              if prediction[i]==2 and forest_target2[i]==3:
                  print(clf.decision_path(forestTest).todense()[i])
          #         print(i)
                  print('b8 is '+str( forest_data2[i][7]))
                  print('\n')
```

```
['b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8', 'b9', 'pred_minus_obs_H_b1', 'pred_minus_obs_H_b2', 'pred_minus_obs_H_b3', 'pr
ed_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', 'pre
d_minus_obs_S_b6', 'pred_minus_obs_S_b7', 'pred_minus_obs_S_b8', 'pred_minus_obs_S_b9']
[[1 0 0 0 0 0 0 0 1 0 0 0 1]]
b8 is 27.0


[[1 0 0 0 0 0 0 0 1 0 0 0 1]]
b8 is 30.0


[[1 0 0 0 0 0 0 0 1 1 0 1 0]]
b8 is 51.0


[[1 0 0 0 0 0 0 0 1 1 0 1 0]]
b8 is 36.0


[[1 0 0 0 0 0 0 0 1 0 0 0 1]]
b8 is 25.0


[[1 0 0 0 0 0 0 0 1 0 0 0 1]]
b8 is 25.0


[[1 0 0 0 0 0 0 0 1 0 0 0 1]]
b8 is 23.0


[[1 0 0 0 0 0 0 0 1 1 0 1 0]]
b8 is 43.0


[[1 0 0 0 0 0 0 0 1 0 0 0 1]]
b8 is 25.0


[[1 0 0 0 0 0 0 0 1 0 0 0 1]]
b8 is 26.0
```

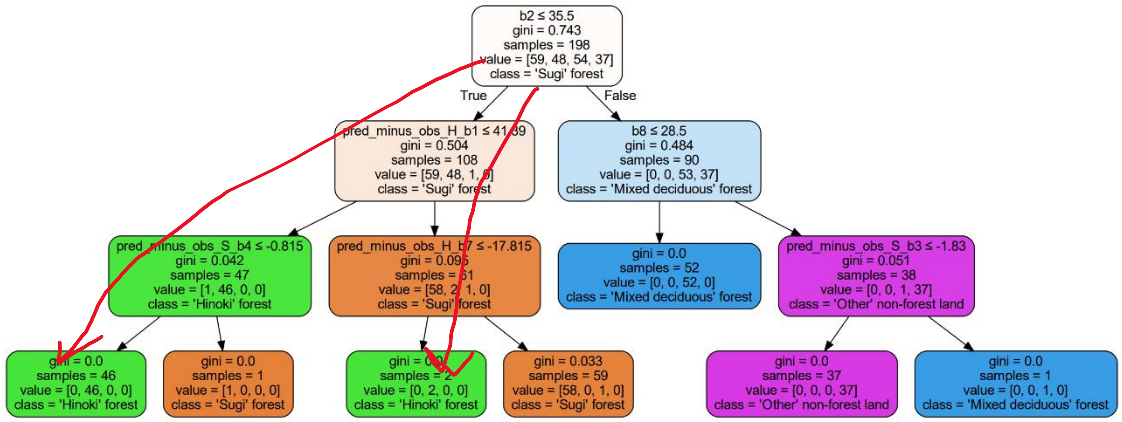
We can see that all the misclassified records with the shorter path has a b8 value <= 28.5. That is the main reason for this misclassification.

“Sugi’ forest” is likely to be misclassified as “Hinoki’ forest”

```
In [182]: #build decision tree 1
          from sklearn import tree
          clf=tree.DecisionTreeClassifier(max_depth=3)
          clf=clf.fit(forest_data,forest_target)

          import numpy as np
          forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
          forestTest= np.array(forest_data2)
          prediction= clf.predict(forestTest)
          # print(forest_target_names[prediction])

          from sklearn.metrics import accuracy_score
          print(accuracy_score(forest_target2,prediction))

          from sklearn.metrics import confusion_matrix
          print(confusion_matrix(forest_target2,prediction))

          0.7846153846153846
          [[119  12   5   0]
           [  9  29   0   0]
           [ 20   0  73  12]
           [  1   1  10  34]]
```

```
In [194]: x=prediction.size
          print(forest_feature_names2)
          for i in range(x):
              if prediction[i]==1 and forest_target2[i]==0:
                  print(clf.decision_path(forestTest).todense()[i])
          #         print(i)
          #         print('b8 is '+str( forest_data2[i][7]))
                  print('\n')

          ['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']
          [[1 1 1 1 0 0 0 0 0 0 0 0 0]]


          [[1 1 1 1 0 0 0 0 0 0 0 0 0]]


          [[1 1 1 1 0 0 0 0 0 0 0 0 0]]


          [[1 1 0 0 0 1 1 0 0 0 0 0 0]]


          [[1 1 0 0 0 1 1 0 0 0 0 0 0]]


          [[1 1 1 1 0 0 0 0 0 0 0 0 0]]


          [[1 1 0 0 0 1 1 0 0 0 0 0 0]]


          [[1 1 0 0 0 1 1 0 0 0 0 0 0]]


          [[1 1 1 1 0 0 0 0 0 0 0 0 0]]


          [[1 1 0 0 0 1 1 0 0 0 0 0 0]]


          [[1 1 0 0 0 1 1 0 0 0 0 0 0]]


          [[1 1 0 0 0 1 1 0 0 0 0 0 0]]
```

We can see the path in all the 12 misclassified records. There are 5 paths consist of node,0,1,2,3, while there are 7 paths consisting of node 0,1,5,6. The path are visualized as below:



For the path consists of node0,1,5,6, the reason behind is due to the overfitting of the model. The leaf node just consists of 2 samples, and therefore causing the misclassification. This situation would have improved when we have pruned this node in tree 5 and tree 6.

For path consist of node 0,1,2,3, the key node for the misclassification is node 1. For all the misclassified records, pred_minus_obs_H_b1<=41.89 gives a true value, which in turn makes the classification as “Hinokoi” instead of “Sugi”. We can verify it by double checking all the misclassified records:

```
In [197]: x=prediction.size
          print(forest_feature_names2)
          for i in range(x):
              if prediction[i]==1 and forest_target2[i]==0:
                  print(clf.decision_path(forestTest).todense()[i])
                 # print(i)
                  print('pred_minus_obs_H_b1 is '+str( forest_data2[i][9]))
                  print('\n')
```

```
['b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8', 'b9', 'pred_minus_obs_H_b1', 'pred_minus_obs_H_b2', 'pred_minus_obs_H_b3', 'pr
ed_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', 'pre
d_minus_obs_S_b6', 'pred_minus_obs_S_b7', 'pred_minus_obs_S_b8', 'pred_minus_obs_S_b9']
[[1 1 1 1 0 0 0 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 37.41


[[1 1 1 1 0 0 0 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 33.48


[[1 1 1 1 0 0 0 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 33.89


[[1 1 0 0 0 1 1 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 45.58


[[1 1 0 0 0 1 1 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 43.06


[[1 1 1 1 0 0 0 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 33.89


[[1 1 0 0 0 1 1 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 49.29


[[1 1 0 0 0 1 1 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 47.57


[[1 1 1 1 0 0 0 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 32.22


[[1 1 0 0 0 1 1 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 45.23


[[1 1 0 0 0 1 1 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 56.28


[[1 1 0 0 0 1 1 0 0 0 0 0 0]]
pred_minus_obs_H_b1 is 65.94
```

We can see that for the 5 records which transversed node 0,1,2,3, they all have pred_minus_obs_H_b1<=41.89 , leading them misclassified as “Hinoki” rather than " Sugi”.

<u>“Hinoki’ forest” is likely to be misclassified as “Sugi’ forest”</u>

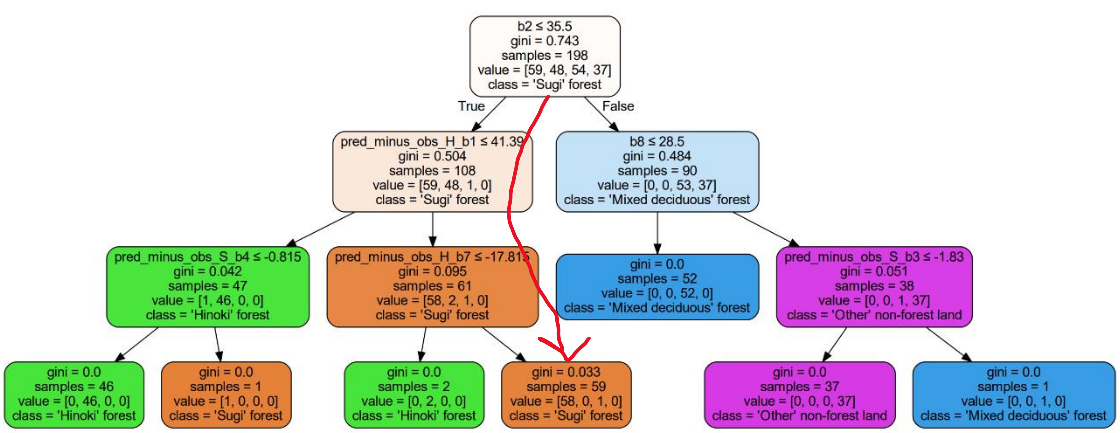
In [182]:
```
#build decision tree 1
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3)
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.7846153846153846
[[119  12   5   0]
 [  9  29   0   0]
 [ 20   0  73  12]
 [  1   1  10  34]]
```

In [198]:
```
x=prediction.size
print(forest_feature_names2)
for i in range(x):
    if prediction[i]==0 and forest_target2[i]==1:
        print(clf.decision_path(forestTest).todense()[i])
       # print(i)
        #print('pred_minus_obs_H_b1 is '+str( forest_data2[i][9]))
        print('\n')
```

```
['b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8', 'b9', 'pred_minus_obs_H_b1', 'pred_minus_obs_H_b2', 'pred_minus_obs_H_b3', 'pr
ed_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', 'pre
d_minus_obs_S_b6', 'pred_minus_obs_S_b7', 'pred_minus_obs_S_b8', 'pred_minus_obs_S_b9']
[[1 1 0 0 0 1 0 1 0 0 0 0 0]]


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]
```

We can see that for “Hinoki” to be misclassified as “Sugi”, the path has to transversed node 0,1,5,7.



Obviously, the key node for the misclassification is pred_minus_obs_H_b1<=41.39. When the result for this node is false, it would be misclassified as "Sugi". We can verify this by checking the pred_minus_obs_H_b1 value in all the misclassified records:

```
In [182]: #build decision tree 1
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3)
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))

0.7846153846153846
[[119  12   5   0]
 [  9  29   0   0]
 [ 20   0  73  12]
 [  1   1  10  34]]
```

```
In [199]: x=prediction.size
print(forest_feature_names2)
for i in range(x):
    if prediction[i]==0 and forest_target2[i]==1:
        print(clf.decision_path(forestTest).todense()[i])
       # print(i)
        print('pred_minus_obs_H_b1 is '+str( forest_data2[i][9]))
        print('\n')

['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']
[[1 1 0 0 0 1 0 1 0 0 0 0 0]]
pred_minus_obs_H_b1 is 43.72


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]
pred_minus_obs_H_b1 is 48.99


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]
pred_minus_obs_H_b1 is 49.53


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]
pred_minus_obs_H_b1 is 59.33


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]
pred_minus_obs_H_b1 is 75.69


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]
pred_minus_obs_H_b1 is 50.66


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]
pred_minus_obs_H_b1 is 44.77


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]
pred_minus_obs_H_b1 is 47.78


[[1 1 0 0 0 1 0 1 0 0 0 0 0]]
pred_minus_obs_H_b1 is 46.95
```

We can see that all records are having pred_minus_obs_H_b1 value >41.39.

(d)(ii). Misclassification in in Tree built using testing.csv

“Hinoki’ forest” is likely to be misclassified as “Sugi’ forest”

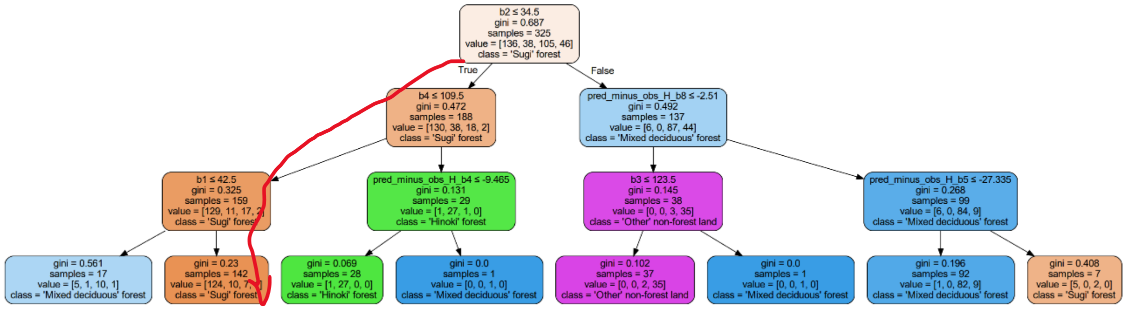
In [208]:
```
#build decision tree 1
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=3)
clf=clf.fit(forest_data,forest_target)

import numpy as np
forest_target_names=["'Sugi' forest","'Hinoki' forest","'Mixed deciduous' forest","'Other' non-forest land"]
forestTest= np.array(forest_data2)
prediction= clf.predict(forestTest)
# print(forest_target_names[prediction])

from sklearn.metrics import accuracy_score
print(accuracy_score(forest_target2,prediction))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(forest_target2,prediction))
```

```
0.8686868686868687
[[55  2  2  0]
 [12 32  4  0]
 [ 0  0 53  1]
 [ 0  0  5 32]]
```

In [209]:
```
x=prediction.size
print(forest_feature_names2)
for i in range(x):
    if prediction[i]==0 and forest_target2[i]==1:
        print(clf.decision_path(forestTest).todense()[i])
       # print(i)
        #print('pred_minus_obs_H_b1 is '+str( forest_data2[i][9]))
        print('\n')
```

```
['b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8', 'b9', 'pred_minus_obs_H_b1', 'pred_minus_obs_H_b2', 'pred_minus_obs_H_b3', 'pr
ed_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', 'pre
d_minus_obs_S_b6', 'pred_minus_obs_S_b7', 'pred_minus_obs_S_b8', 'pred_minus_obs_S_b9']
[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
```

We can see that all the misclassified records go through node 0,1, 2, 4.



The key node for misclassification is b4<=109.5. When this node gives a true value, misclassification would happen. We could verify this conclusion looking at the values in the 12 misclassified records:

In [214]:
```python
x=prediction.size
print(forest_feature_names2)
for i in range(x):
    if prediction[i]==0 and forest_target2[i]==1:
        print(clf.decision_path(forestTest).todense()[i])
       # print(i)
        print('b4 is '+str( forest_data2[i][3])+'pred_minus_obs_S_b4 is '+str( forest_data2[i][21]))
        print('\n')
```

```
['b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8', 'b9', 'pred_minus_obs_H_b1', 'pred_minus_obs_H_b2', 'pred_minus_obs_H_b3', 'pr
ed_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', 'pre
d_minus_obs_S_b6', 'pred_minus_obs_S_b7', 'pred_minus_obs_S_b8', 'pred_minus_obs_S_b9']
[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 109.0pred_minus_obs_S_b4 is -18.88


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 104.0pred_minus_obs_S_b4 is -10.57


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 107.0pred_minus_obs_S_b4 is -10.48


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 102.0pred_minus_obs_S_b4 is -3.51


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 106.0pred_minus_obs_S_b4 is -25.68


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 109.0pred_minus_obs_S_b4 is -16.02


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 107.0pred_minus_obs_S_b4 is -6.14


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 99.0pred_minus_obs_S_b4 is -12.15


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 109.0pred_minus_obs_S_b4 is -17.39


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 109.0pred_minus_obs_S_b4 is -23.18


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 100.0pred_minus_obs_S_b4 is -17.45


[[1 1 1 0 1 0 0 0 0 0 0 0 0 0 0]]
b4 is 106.0pred_minus_obs_S_b4 is -7.12
```

We could see that all the b4 values <=109.5, making the decision fall into the left branch instead of the right branch.

If b4 values of all these misclassifications are made greater than 109.5, it still doesn’t mean that the classification can be corrected done in tree 1. Next thing we need to check is pred_minus_obs_S_b4 <= -9.465. From the values displayed in the above image, it is shown that 3 records will still be classified as “Mixed” as pred_minus_obs_S_b4 >-9.465. However, this problem should have been resolved when the leaf node are pruned starting from tree 3.