We start with by uploading the data and defining labels for Symptoms and the Diagnosis.

In [62]:
```
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.datasets import load_iris
from sklearn.datasets import load_breast_cancer
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
import pandas as pd
import numpy as np
from sklearn import tree
```

In [77]:
```
dataset = pd.read_csv("C:\\Users\\Akanksha\\Downloads\\sledata.txt", sep = " ", header = None)

columns = ['S1','S2','S3','S4','S5','S6','S7','S8','S9','S10','S11','D']
dataset.columns = columns
dataset['D'] = dataset['D'].astype('string')
dataset
```

Out[77]:

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	D
0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	1
2	0	0	0	0	1	0	0	0	0	0	0	1
3	0	0	0	1	0	0	0	0	0	0	0	1
4	0	0	0	0	1	0	0	0	1	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
295	1	0	0	0	1	0	0	0	1	1	1	2
296	1	0	1	1	0	0	0	0	1	1	1	2
297	0	0	0	1	1	1	0	0	0	1	1	2
298	1	0	1	0	1	1	1	1	0	1	1	2
299	1	0	1	1	1	1	0	1	1	1	1	2

300 rows × 12 columns

After loading the data, we divide the data into the dependent and independent variables. X has all the symptoms and Y has the diagnosis.

```
In [58]: X = dataset[['S1','S2','S3','S4','S5','S6','S7','S8','S9','S10','S11']].values
         X[0:5]

Out[58]: array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1],
                [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
                [0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1]], dtype=int64)

In [59]: y = dataset["D"]
         y[0:5]

Out[59]: 0    1
         1    1
         2    1
         3    1
         4    1
         Name: D, dtype: string
```

We then divide the data into training and testing datasets. I have chosen 30% of the dataset to be dedicated to testing our model.

```
In [100]: from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=3)
```

I am going to create the tree using the DecisionTreeClassifier within sklearn in these combinations to test the best accuracy –

1. **Use the Entropy criterion first without defining a maximum depth.**
2. **Use the Gini criterion first without defining a maximum depth.**
3. **Use the Entropy criterion first with setting a max_depth of 3.**
4. **Use the Gini criterion first with setting a max_depth of 3.**

1. Use the Entropy criterion first without defining a maximum depth.

```
In [19]: from sklearn.tree import DecisionTreeClassifier
         classifier = DecisionTreeClassifier(criterion= 'entropy')
         classifier.fit(X_train, y_train)

Out[19]: DecisionTreeClassifier(criterion='entropy')
```

The accuracy of this tree is 95.5% and below is the confusion matrix with only four incorrect classifications out of 90 classifications *(our test data set was 30% of 300 rows, i.e., 90 rows of data on which this model was tested).*

We used the accuracy score method within sklearn to get the accuracy of our decision tree.

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = classifier.predict(X_test)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_test, y_pred))
```

```
DecisionTrees's Accuracy:  0.9555555555555556
```

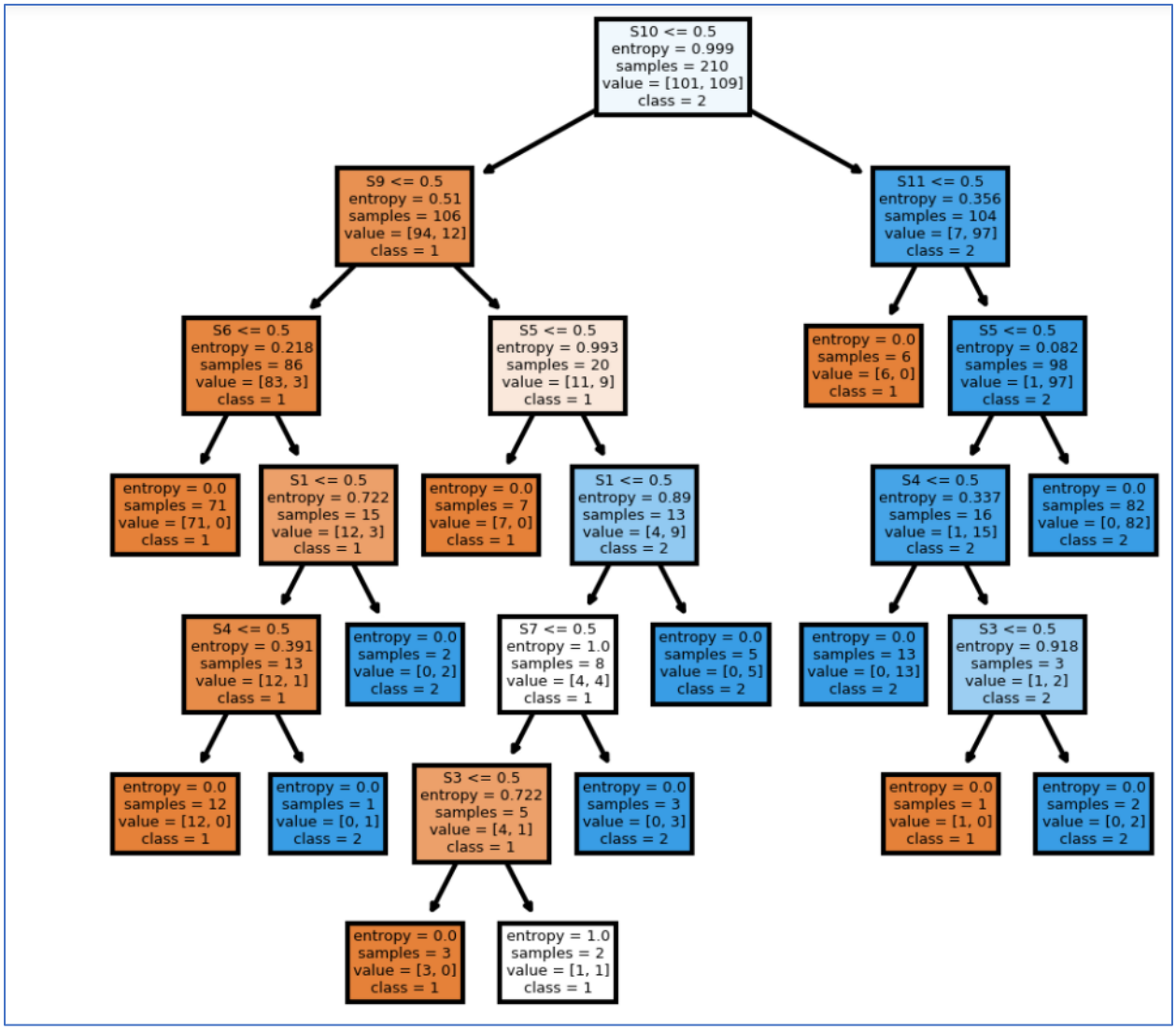
Confusion Matrix – Correct Classifications -> 86/90 = 95.55 %

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[49  0]
 [ 4 37]]
              precision    recall  f1-score   support

           1       0.92      1.00      0.96        49
           2       1.00      0.90      0.95        41

    accuracy                           0.96        90
   macro avg       0.96      0.95      0.95        90
weighted avg       0.96      0.96      0.96        90
```

We then proceed to plot the decision tree to look at the splits, entropy and sample size of every node.

```
fn=['S1','S2','S3','S4','S5','S6','S7','S8','S9','S10','S11','D']
cn=['1','2']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
tree.plot_tree(clf,
               feature_names = fn,
               class_names=cn,
               filled = True);
fig.savefig('imagename.png')
```



We can see that this tree has a high depth(6) as we did not limit the maximum depth parameter while plotting the tree. Moving on, we will limit the depth and see how that affects accuracy.

There are a total of 27 nodes in this tree.

Also, we should note that the leaf nodes very low samples/cases (less than 10 or 20).

```
print("Depth of the Decision tree is:",tree1.tree_.max_depth)
print("Total number of nodes in the tree are",tree1.tree_.node_count)
```

```
Depth of the Decision tree is: 6
Total number of nodes in the tree are 27
```

2. Use the Gini criterion without defining a maximum depth -

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion= 'gini')
clf.fit(X_train, y_train)
```

```
62]: DecisionTreeClassifier()
```

We see that the accuracy is slightly higher than entropy criterion we used above. The accuracy of the decision tree built on the Gini criterion is 96.66 % (Vs. 95.55% of Entropy criterion-based decision Tree)

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_test)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_test, y_pred))
```

```
DecisionTrees's Accuracy:  0.9666666666666667
```

We have one more accurate classification on the same test dataset from above, see below
(No. of misclassification = 3 for Gini Vs. 4 for Entropy Decision tree; Gini is slightly better)

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

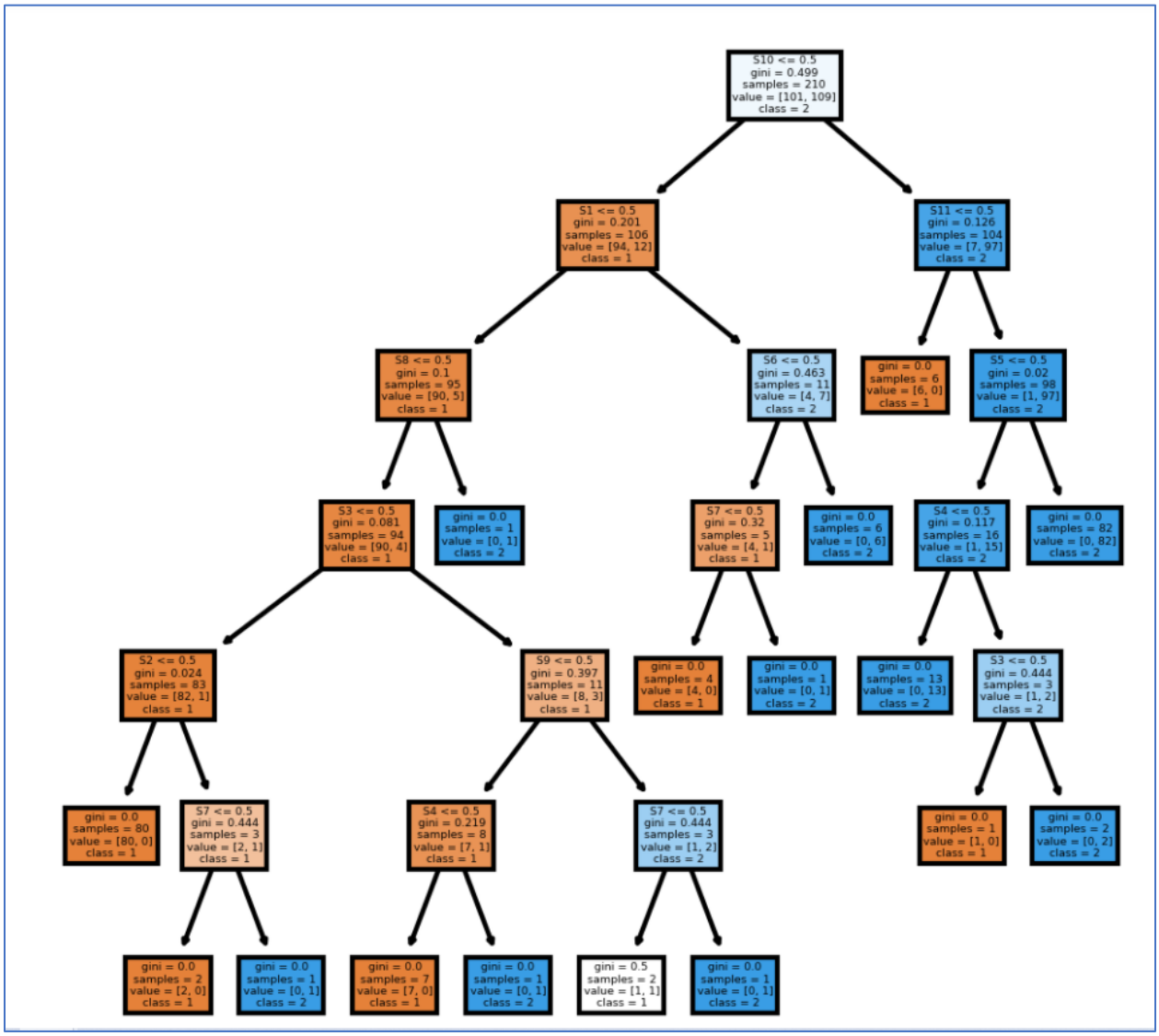
```
[[49  0]
 [ 3 38]]
              precision    recall  f1-score   support

           1       0.94      1.00      0.97        49
           2       1.00      0.93      0.96        41

    accuracy                           0.97        90
   macro avg       0.97      0.96      0.97        90
weighted avg       0.97      0.97      0.97        90
```

However, while plotting the tree we see that the depth of the tree is the same as compared to the previous tree (entropy based) but the number of nodes has increased by 4 from 27 to 31.

```
print("Depth of the Decision tree is:",tree1.tree_.max_depth)
print("Total number of nodes in the tree are",tree1.tree_.node_count)
```

```
Depth of the Decision tree is: 6
Total number of nodes in the tree are 31
```

Also, we should note that the leaf nodes still has very low samples/cases (less than 10 or 20).



3. **Use the Entropy criterion first with setting a max_depth of 3– When we limit the maximum depth of the tree to 3, the accuracy falls slightly from 95.5% to 93.33%. See below –**

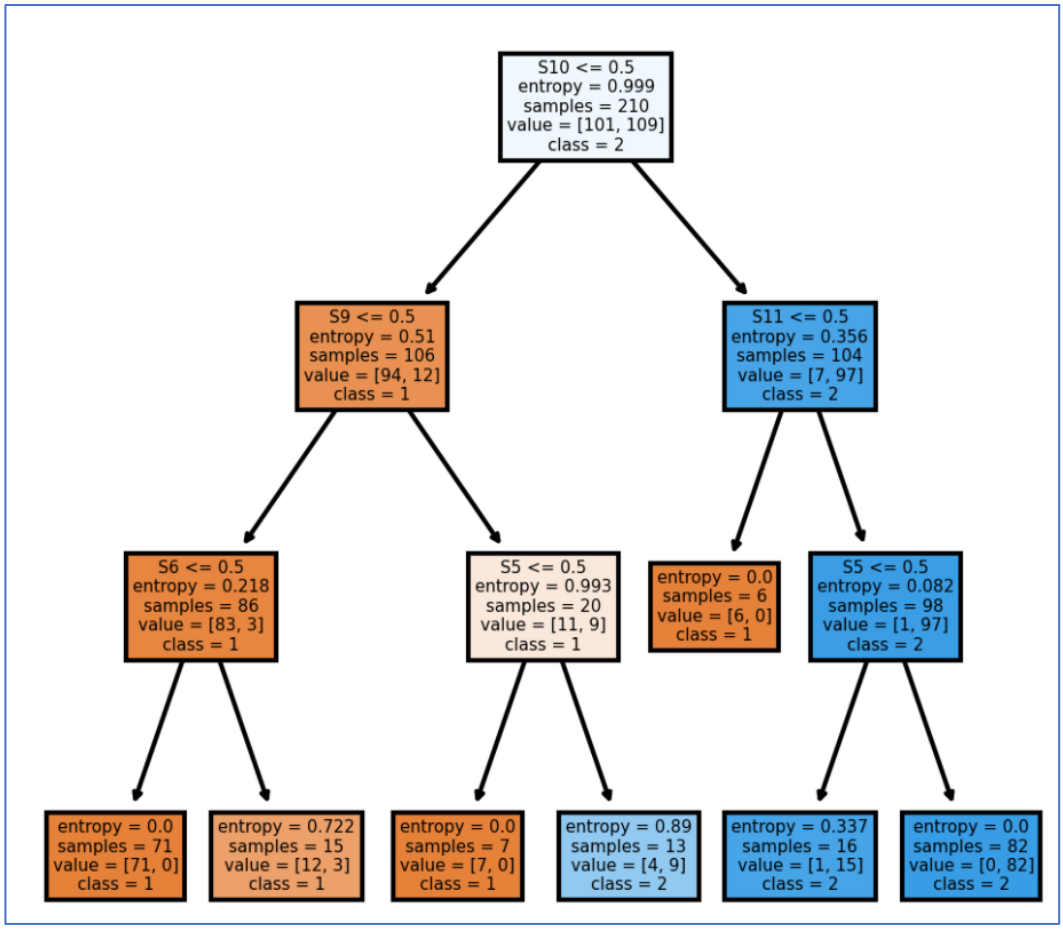
```
In [49]: from sklearn.tree import DecisionTreeClassifier
         clf = DecisionTreeClassifier(criterion= 'entropy', max_depth = 3)
         clf.fit(X_train, y_train)

Out[49]: DecisionTreeClassifier(criterion='entropy', max_depth=3)

In [50]: from sklearn import metrics
         import matplotlib.pyplot as plt
         y_pred = clf.predict(X_test)
         print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_test, y_pred))

         DecisionTrees's Accuracy:  0.9333333333333333
```

No. of misclassifications went up from 4 to 6 when we limited the depth of the tree to 3.

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[45  4]
 [ 2 39]]
              precision    recall  f1-score   support

           1       0.96      0.92      0.94        49
           2       0.91      0.95      0.93        41

    accuracy                           0.93        90
   macro avg       0.93      0.93      0.93        90
weighted avg       0.93      0.93      0.93        90
```

We can see that the depth of the tree has been reduced but the parent node is still S10 with the same number of samples 210.



4. **Use the Gini criterion first with setting a max_depth of 3– When we limit the maximum depth of the tree to 3, the accuracy falls slightly from 96.66% to 94.44% and the total number of nodes reduces . See below –**

```
In [88]: from sklearn.tree import DecisionTreeClassifier
         clf = DecisionTreeClassifier(criterion= 'gini', max_depth =3)
         clf.fit(X_train, y_train)

Out[88]: DecisionTreeClassifier(max_depth=3)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_test)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_test, y_pred))
```

```
DecisionTrees's Accuracy:  0.9444444444444444
```

No of incorrect classifications go up to 5 from 3 when we did not limit depth using Gini earlier.

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[49  0]
 [ 5 36]]
              precision    recall  f1-score   support

           1       0.91      1.00      0.95        49
           2       1.00      0.88      0.94        41

    accuracy                           0.94        90
   macro avg       0.95      0.94      0.94        90
weighted avg       0.95      0.94      0.94        90
```

The number of nodes also decreased by half when limiting depth

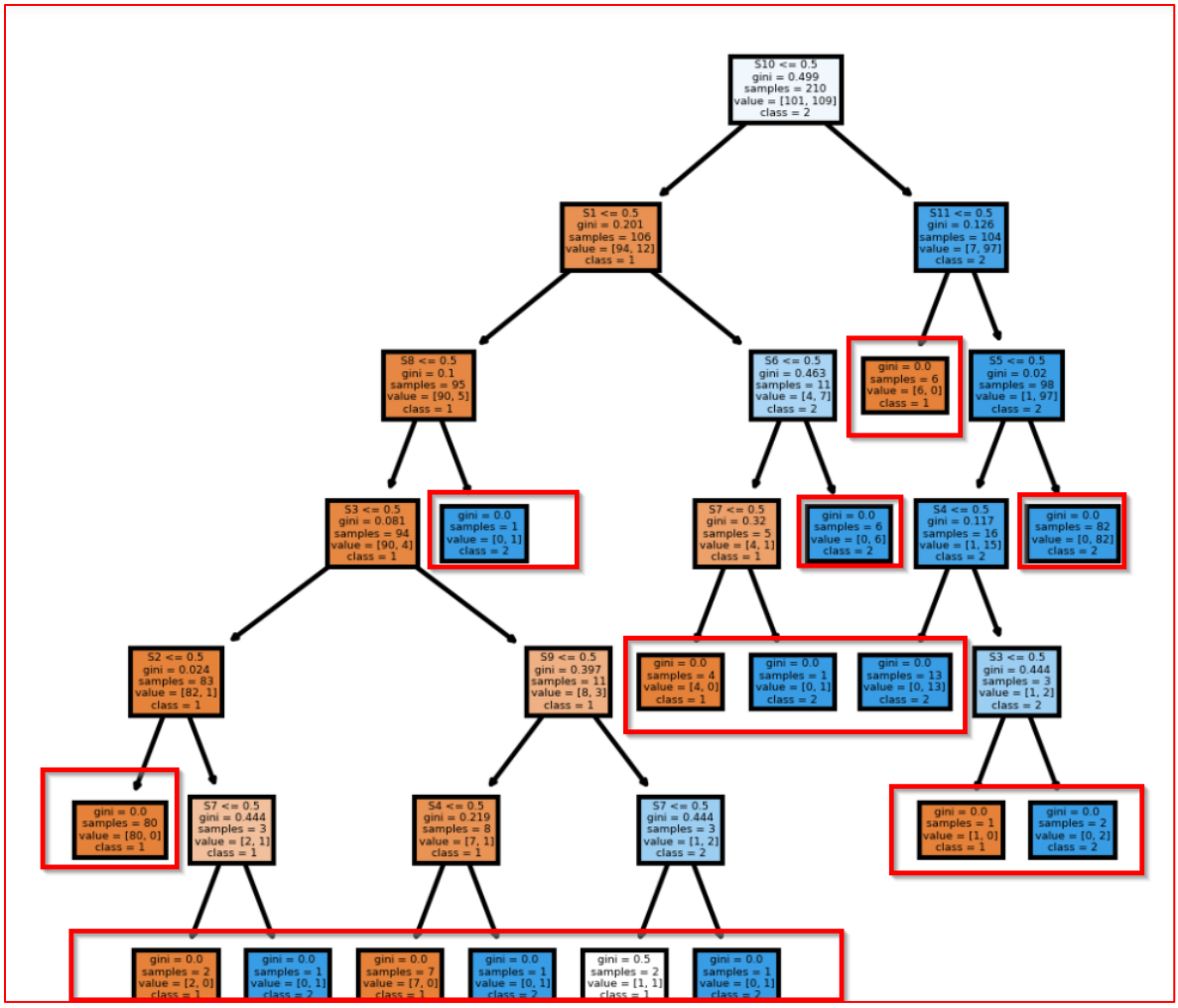
```
print("Depth of the Decision tree is:",tree1.tree_.max_depth)
print("Total number of nodes in the tree are",tree1.tree_.node_count)
```

```
Depth of the Decision tree is: 3
Total number of nodes in the tree are 13
```

Looking at all these combinations, I would choose the Decision Tree made on the Gini criterion as it has the highest accuracy of 96.66%.

2) How many nodes the final tree has and how many of them are terminal nodes?

The tree which we have chosen has 31 nodes and 16 terminal nodes *(in red below)*


S10 <= 0.5
gini = 0.499
samples = 210
value = [101, 109]
class = 2
S1 <= 0.5
gini = 0.201
samples = 106
value = [94, 12]
class = 1
S11 <= 0.5
gini = 0.126
samples = 104
value = [7, 97]
class = 2
S8 <= 0.5
gini = 0.1
samples = 95
value = [90, 5]
class = 1
S6 <= 0.5
gini = 0.463
samples = 11
value = [4, 7]
class = 2
gini = 0.0
samples = 6
value = [6, 0]
class = 1
S5 <= 0.5
gini = 0.02
samples = 98
value = [1, 97]
class = 2
S3 <= 0.5
gini = 0.081
samples = 94
value = [90, 4]
class = 1
gini = 0.0
samples = 1
value = [0, 1]
class = 2
S7 <= 0.5
gini = 0.32
samples = 5
value = [4, 1]
class = 1
gini = 0.0
samples = 6
value = [0, 6]
class = 2
S4 <= 0.5
gini = 0.117
samples = 16
value = [1, 15]
class = 2
gini = 0.0
samples = 82
value = [0, 82]
class = 2
S2 <= 0.5
gini = 0.024
samples = 83
value = [82, 1]
class = 1
S9 <= 0.5
gini = 0.397
samples = 11
value = [8, 3]
class = 1
gini = 0.0
samples = 4
value = [4, 0]
class = 1
gini = 0.0
samples = 1
value = [0, 1]
class = 2
gini = 0.0
samples = 13
value = [0, 13]
class = 2
S3 <= 0.5
gini = 0.444
samples = 3
value = [1, 2]
class = 2
gini = 0.0
samples = 80
value = [80, 0]
class = 1
S7 <= 0.5
gini = 0.444
samples = 3
value = [2, 1]
class = 1
S4 <= 0.5
gini = 0.219
samples = 8
value = [7, 1]
class = 1
S7 <= 0.5
gini = 0.444
samples = 3
value = [1, 2]
class = 2
gini = 0.0
samples = 1
value = [1, 0]
class = 1
gini = 0.0
samples = 2
value = [0, 2]
class = 2
gini = 0.0
samples = 2
value = [2, 0]
class = 1
gini = 0.0
samples = 1
value = [0, 1]
class = 2
gini = 0.0
samples = 7
value = [7, 0]
class = 1
gini = 0.0
samples = 1
value = [0, 1]
class = 2
gini = 0.5
samples = 2
value = [1, 1]
class = 1
gini = 0.0
samples = 1
value = [0, 1]
class = 2

3) What are the most important three Lupus data features in building the tree? Explain your answer.

The three most important lupus data features are Symptom 10, Symptom 11 and Symptom 1 as they have the highest normalized feature importance score in the decision tree.

These importance values have been calculated using the features_importance method within sklearn, which compute the importance of features based on total reduction of the criterion brought by that feature. It is also known as the Gini importance.

Please see below, features sorted based on their importance:

Feature	Importance
'S10'	0.67894583
'S11'	0.1066765
'S1'	0.06469504
'S6'	0.03361569
'S9'	0.02773586
'S3'	0.02555059
'S4'	0.02206759
'S8'	0.01746896
'S7'	0.01604916
'S2'	0.00618763
'S5'	0.00100717

4) Increase the number of cases for each parent and child. What do you notice with the complexity of the tree? Does it increase? Explain your answer.

In our original decision tree (Gini, Depth = 6), we can see that most leaf nodes have a sample size which is less than 20. When we increase the minimum sample size of nodes to 20, we see that the complexity of the tree decreases along with the following changes to the decision tree:

- **Total number of nodes decreases from 31 to 13.**
- **The depth of the tree reduces from 6 to 3.**
- **However, the accuracy reduces from 96.66% to 90%.**

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion= 'gini',min_samples_leaf = 20)
clf.fit(X_train, y_train)
```

```
6]: DecisionTreeClassifier(min_samples_leaf=20)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt
y_pred = clf.predict(X_test)
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_test, y_pred))
```

```
DecisionTrees's Accuracy:  0.9
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[46  3]
 [ 6 35]]
              precision    recall  f1-score   support

           1       0.88      0.94      0.91        49
           2       0.92      0.85      0.89        41

    accuracy                           0.90        90
   macro avg       0.90      0.90      0.90        90
weighted avg       0.90      0.90      0.90        90
```

New Tree after increasing sample size in nodes: