```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import tree
```

```
import pandas as pd
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target #flower cate..
```

```
df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

```
X_train, X_test, Y_train, Y_test = train_test_split(df[iris.feature_names], df['target'],
print(len( X_train))
```

```
112
```

```
clf = DecisionTreeClassifier(criterion='entropy') #gini index
```

```
clf.fit(X_train, Y_train) #training the model
```

```
DecisionTreeClassifier(criterion='entropy')
```

```
print(X_test)
Y_pred=clf.predict(X_test)
```

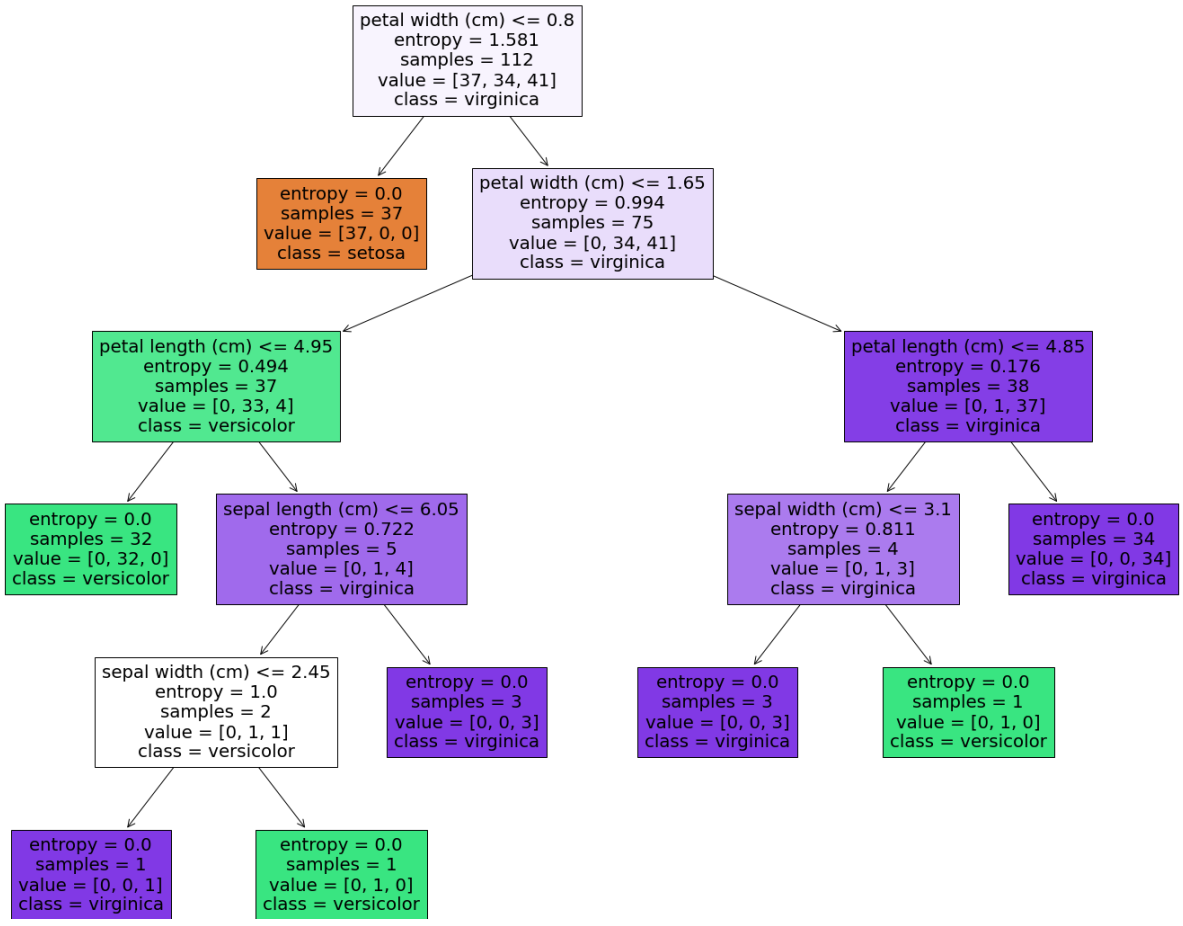
```
print(Y_pred)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
14                 5.8               4.0                1.2               0.2
98                 5.1               2.5                3.0               1.1
75                 6.6               3.0                4.4               1.4
16                 5.4               3.9                1.3               0.4
131                7.9               3.8                6.4               2.0
56                 6.3               3.3                4.7               1.6
141                6.9               3.1                5.1               2.3
44                 5.1               3.8                1.9               0.4
29                 4.7               3.2                1.6               0.2
120                6.9               3.2                5.7               2.3
94                 5.6               2.7                4.2               1.3
5                  5.4               3.9                1.7               0.4
102                7.1               3.0                5.9               2.1
51                 6.4               3.2                4.5               1.5
78                 6.0               2.9                4.5               1.5
42                 4.4               3.2                1.3               0.2
92                 5.8               2.6                4.0               1.2
66                 5.6               3.0                4.5               1.5
31                 5.4               3.4                1.5               0.4
35                 5.0               3.2                1.2               0.2
90                 5.5               2.6                4.4               1.2
84                 5.4               3.0                4.5               1.5
77                 6.7               3.0                5.0               1.7
40                 5.0               3.5                1.3               0.3
125                7.2               3.2                6.0               1.8
99                 5.7               2.8                4.1               1.3
33                 5.5               4.2                1.4               0.2
19                 5.1               3.8                1.5               0.3
73                 6.1               2.8                4.7               1.2
146                6.3               2.5                5.0               1.9
91                 6.1               3.0                4.6               1.4
135                7.7               3.0                6.1               2.3
69                 5.6               2.5                3.9               1.1
128                6.4               2.8                5.6               2.1
114                5.8               2.8                5.1               2.4
48                 5.3               3.7                1.5               0.2
53                 5.5               2.3                4.0               1.3
28                 5.2               3.4                1.4               0.2
[0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 2 0 2 1 0 0 1 2 1 2 1 2 2 0 1
 0]
```

```
fig = plt.figure(figsize=(25,20))
tree.plot_tree(clf,

feature_names=iris.feature_names,
class_names=iris.target_names,
filled=True)
```

```
[Text(0.4, 0.9166666666666666, 'petal width (cm) <= 0.8\nentropy = 1.581\nsamples
= 112\nvalue = [37, 34, 41]\nclass = virginica'),
 Text(0.3, 0.75, 'entropy = 0.0\nsamples = 37\nvalue = [37, 0, 0]\nclass =
setosa'),
 Text(0.5, 0.75, 'petal width (cm) <= 1.65\nentropy = 0.994\nsamples = 75\nvalue =
[0, 34, 41]\nclass = virginica'),
 Text(0.2, 0.5833333333333334, 'petal length (cm) <= 4.95\nentropy =
0.494\nsamples = 37\nvalue = [0, 33, 4]\nclass = versicolor'),
 Text(0.1, 0.4166666666666667, 'entropy = 0.0\nsamples = 32\nvalue = [0, 32,
0]\nclass = versicolor'),
 Text(0.3, 0.4166666666666667, 'sepal length (cm) <= 6.05\nentropy =
0.722\nsamples = 5\nvalue = [0, 1, 4]\nclass = virginica'),
 Text(0.2, 0.25, 'sepal width (cm) <= 2.45\nentropy = 1.0\nsamples = 2\nvalue =
[0, 1, 1]\nclass = versicolor'),
 Text(0.1, 0.08333333333333333, 'entropy = 0.0\nsamples = 1\nvalue = [0, 0,
1]\nclass = virginica'),
 Text(0.3, 0.08333333333333333, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1,
0]\nclass = versicolor'),
 Text(0.4, 0.25, 'entropy = 0.0\nsamples = 3\nvalue = [0, 0, 3]\nclass =
virginica'),
 Text(0.8, 0.5833333333333334, 'petal length (cm) <= 4.85\nentropy =
0.176\nsamples = 38\nvalue = [0, 1, 37]\nclass = virginica'),
 Text(0.7, 0.4166666666666667, 'sepal width (cm) <= 3.1\nentropy = 0.811\nsamples
= 4\nvalue = [0, 1, 3]\nclass = virginica'),
 Text(0.6, 0.25, 'entropy = 0.0\nsamples = 3\nvalue = [0, 0, 3]\nclass =
virginica'),
 Text(0.8, 0.25, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass =
versicolor'),
 Text(0.9, 0.4166666666666667, 'entropy = 0.0\nsamples = 34\nvalue = [0, 0,
34]\nclass = virginica')]
```



```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(Y_test, Y_pred))
print(classification_report(Y_test, Y_pred))
```

```
[[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      0.94      0.97        16
           2       0.90      1.00      0.95         9

    accuracy                           0.97        38
   macro avg       0.97      0.98      0.97        38
weighted avg       0.98      0.97      0.97        38
```

```
from sklearn import metrics
acc= metrics.accuracy_score(Y_test, Y_pred)
print("Accuracy:",acc)
```

```
Accuracy: 0.9736842105263158
```