In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [3]:
```
data = pd.read_csv('C:\\Iris.csv')
data.head()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [5]:
```
data['Species'].value_counts()
```

Out[5]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

In [6]:
```
data.drop('Id', axis=1, inplace=True)
```

In [7]:
```
data['Species'].value_counts().to_dict()
```

Out[7]:
```
{'Iris-setosa': 50, 'Iris-versicolor': 50, 'Iris-virginica': 50}
```

In [8]:
```
data['Species'].replace({'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2},
```

In [9]:
```
data.head()
```

Out[9]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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

In [10]:
```
data['Species'].value_counts()
```

Out[10]:
```
Species
0    50
1    50
2    50
Name: count, dtype: int64
```

In [11]:
```
X = data.drop('Species', axis=1)
X.head()
```

Out[11]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [12]:
```
Y = data['Species']
Y
```

Out[12]:
```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: Species, Length: 150, dtype: int64
```

In [13]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, random_state=8, stratify=Y, t
```

In [14]:
```
X_train
```

Out[14]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
117	7.7	3.8	6.7	2.2
145	6.7	3.0	5.2	2.3
138	6.0	3.0	4.8	1.8
55	5.7	2.8	4.5	1.3
35	5.0	3.2	1.2	0.2
...	...	...	...	...
53	5.5	2.3	4.0	1.3
83	6.0	2.7	5.1	1.6
127	6.1	3.0	4.9	1.8
146	6.3	2.5	5.0	1.9
67	5.8	2.7	4.1	1.0

120 rows × 4 columns

In [15]:
```
Y_train
```

Out[15]:
```
117    2
145    2
138    2
55     1
35     0
      ..
53     1
83     1
127    2
146    2
67     1
Name: Species, Length: 120, dtype: int64
```

In [16]:
```
X_test
```

Out[16]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
133	6.3	2.8	5.1	1.5
41	4.5	2.3	1.3	0.3
20	5.4	3.4	1.7	0.2
116	6.5	3.0	5.5	1.8
135	7.7	3.0	6.1	2.3
86	6.7	3.1	4.7	1.5
141	6.9	3.1	5.1	2.3
10	5.4	3.7	1.5	0.2
132	6.4	2.8	5.6	2.2
77	6.7	3.0	5.0	1.7
75	6.6	3.0	4.4	1.4
115	6.4	3.2	5.3	2.3
110	6.5	3.2	5.1	2.0
54	6.5	2.8	4.6	1.5
88	5.6	3.0	4.1	1.3
40	5.0	3.5	1.3	0.3
99	5.7	2.8	4.1	1.3
57	4.9	2.4	3.3	1.0
87	6.3	2.3	4.4	1.3
143	6.8	3.2	5.9	2.3
5	5.4	3.9	1.7	0.4
26	5.0	3.4	1.6	0.4
21	5.1	3.7	1.5	0.4
3	4.6	3.1	1.5	0.2
81	5.5	2.4	3.7	1.0
19	5.1	3.8	1.5	0.3
8	4.4	2.9	1.4	0.2
128	6.4	2.8	5.6	2.1
102	7.1	3.0	5.9	2.1
52	6.9	3.1	4.9	1.5

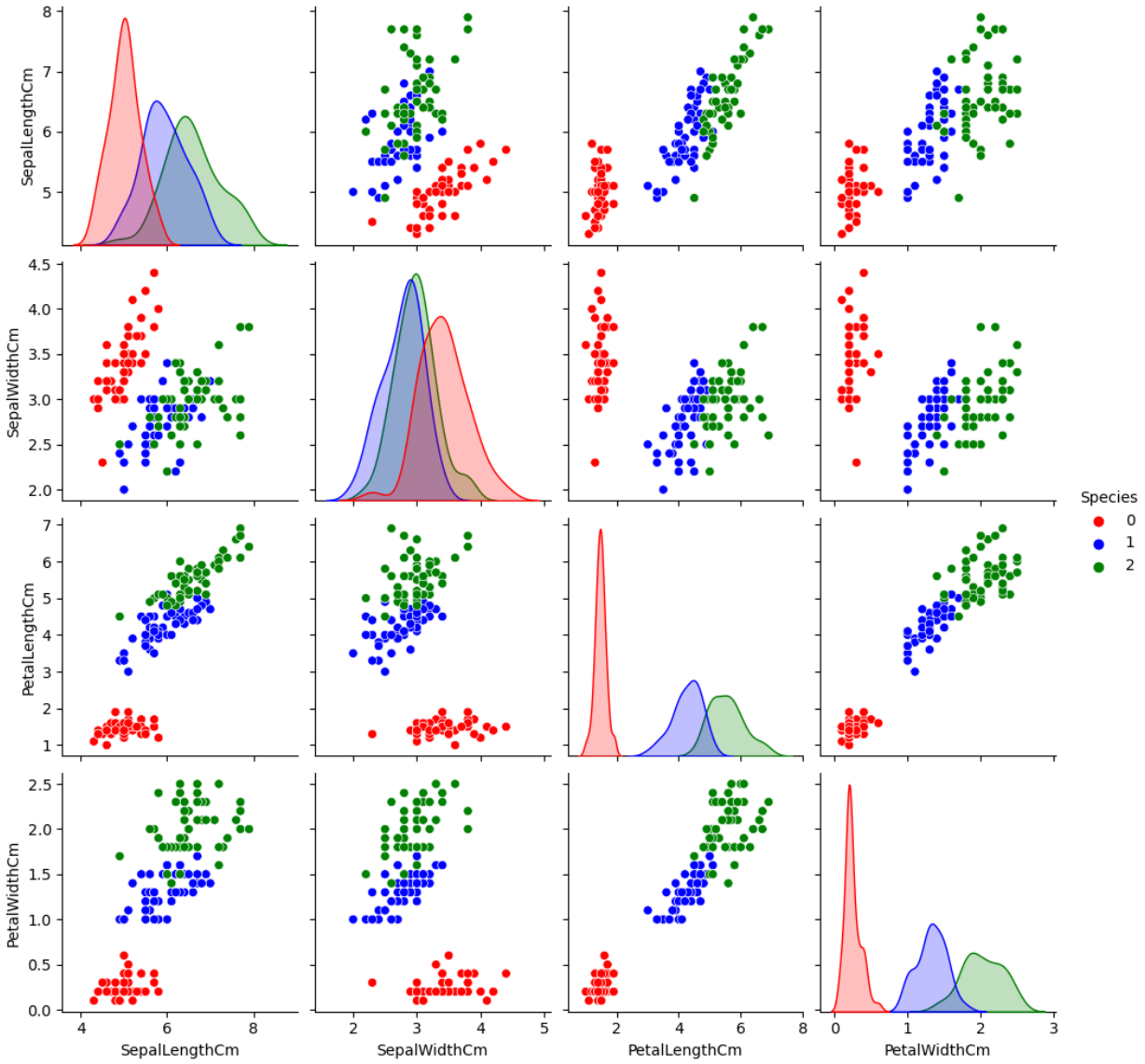
In [17]:
```
Y_test
```

Out[17]:
```
133    2
41     0
20     0
116    2
135    2
86     1
141    2
10     0
132    2
77     1
75     1
115    2
110    2
54     1
88     1
40     0
99     1
57     1
87     1
143    2
5      0
26     0
21     0
3      0
81     1
19     0
8      0
128    2
102    2
52     1
Name: Species, dtype: int64
```

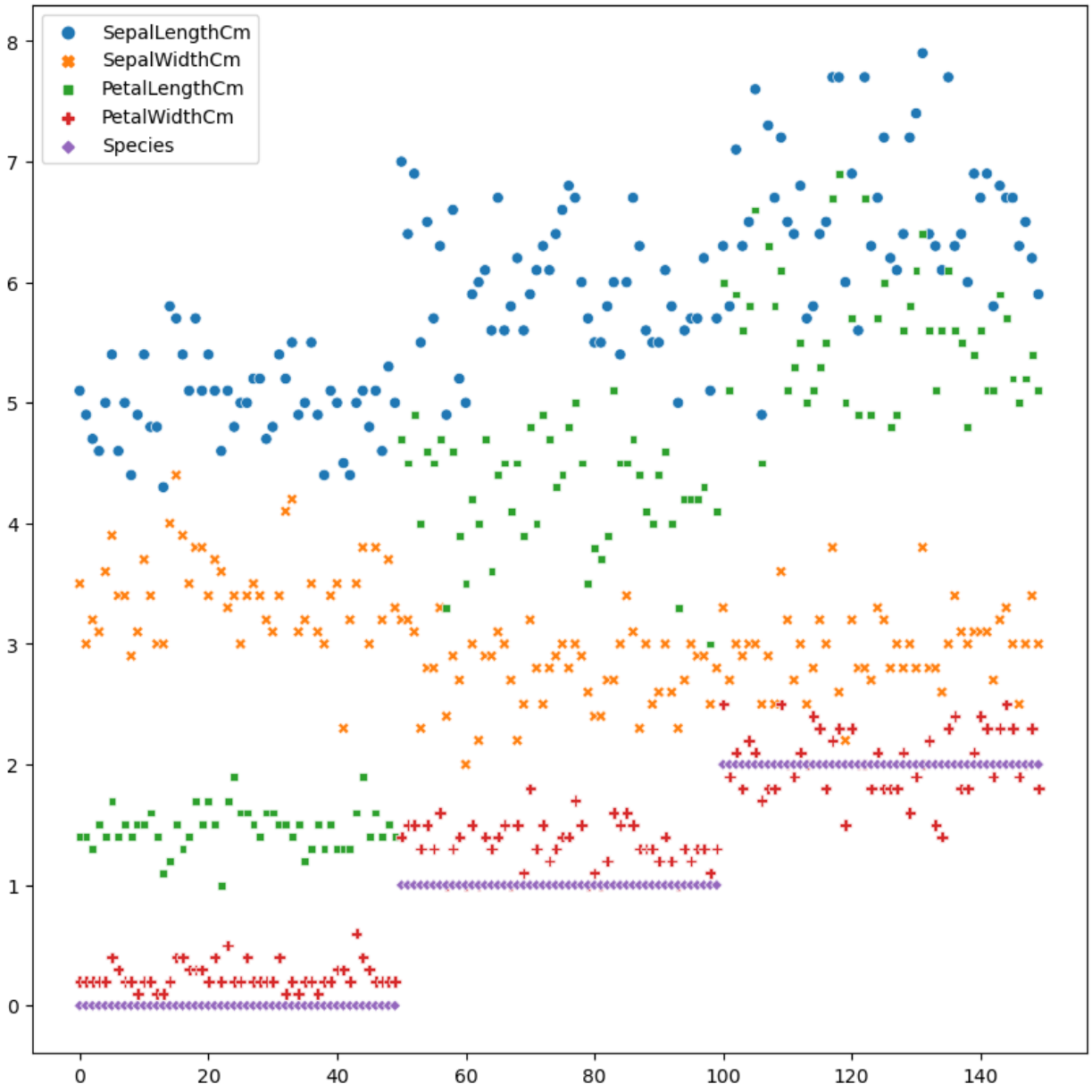
In [18]:
```
plt.figure(figsize=(10,12))
sns.pairplot(data, hue='Species', palette=['Red', 'Blue', 'Green'])
```

```
C:\Users\admin\Documents\Arduino\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x14903ea9310>

<Figure size 1000x1200 with 0 Axes>
```



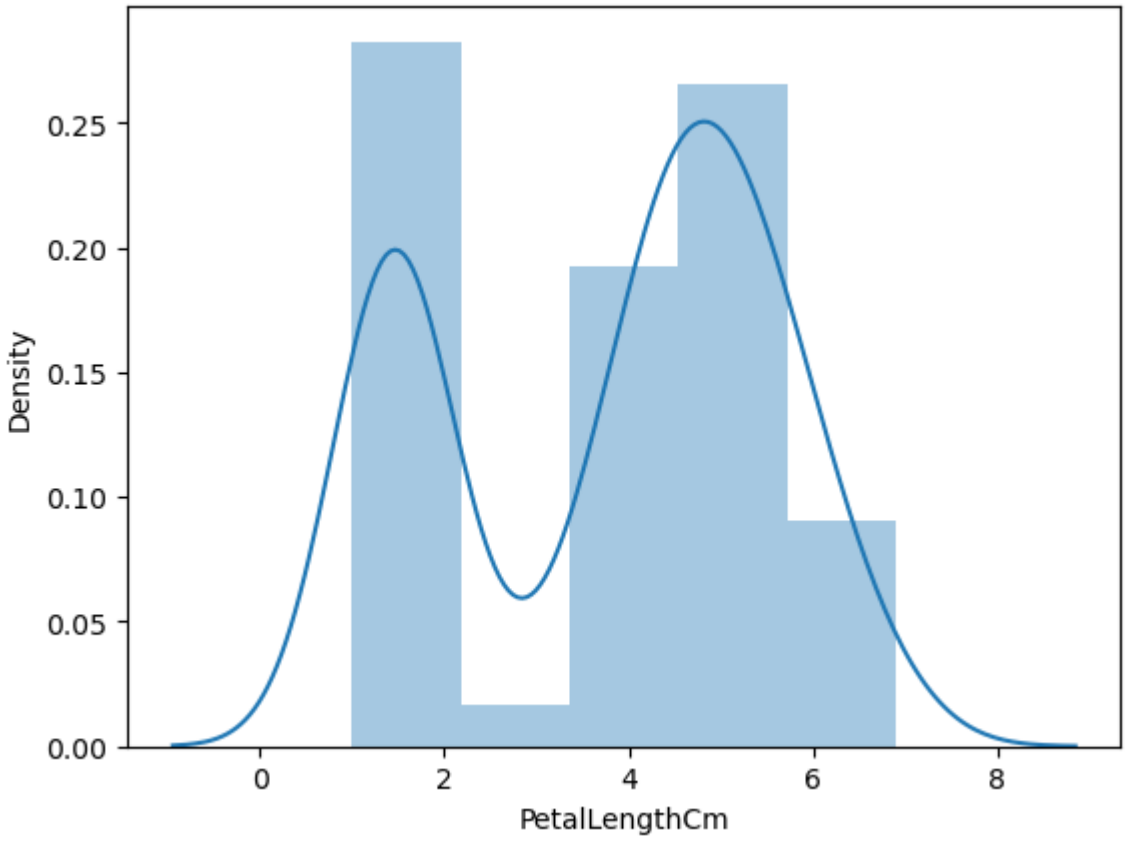
In [19]:
```
plt.figure(figsize=(10,10))
sns.scatterplot(data)
plt.show()
```



In [20]:
```
sns.distplot(data['PetalLengthCm'])
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_1884\4073005315.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['PetalLengthCm'])
```

Out[20]:
```
<Axes: xlabel='PetalLengthCm', ylabel='Density'>
```



In [21]:
```
dt_model = DecisionTreeClassifier()
dt_model
```

Out[21]:
```
DecisionTreeClassifier()
```

In [22]:
```
dt_model.fit(X_train, Y_train)
```

Out[22]:
```
DecisionTreeClassifier()
```

In [23]:
```
Y_prediction_test = dt_model.predict(X_test)
Y_prediction_test
```

Out[23]:
```
array([2, 0, 0, 2, 2, 1, 2, 0, 2, 2, 1, 2, 2, 1, 1, 0, 1, 1, 1, 2, 0, 0,
       0, 0, 1, 0, 0, 2, 2, 1], dtype=int64)
```

In [25]:
```
accuracy_train = accuracy_score(Y_prediction_train, Y_train)*100
accuracy_train
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[25], line 1
----> 1 accuracy_train = accuracy_score(Y_prediction_train, Y_train)*100
      2 accuracy_train

NameError: name 'Y_prediction_train' is not defined
```

In [26]:
```
Y_prediction_train = dt_model.predict(X_train)
Y_prediction_train
```

Out[26]:
```
array([2, 2, 2, 1, 0, 0, 2, 2, 1, 2, 0, 0, 0, 0, 0, 2, 1, 2, 1, 0, 1, 1,
       1, 2, 2, 0, 0, 2, 1, 0, 2, 0, 0, 0, 1, 0, 1, 0, 2, 0, 1, 1, 1, 0,
       2, 0, 2, 1, 2, 1, 1, 1, 1, 0, 0, 2, 0, 1, 2, 0, 2, 0, 1, 1, 0, 0,
       2, 0, 0, 0, 0, 0, 1, 2, 1, 0, 2, 0, 2, 2, 1, 0, 1, 2, 1, 2, 2, 0,
       2, 0, 1, 0, 1, 2, 1, 2, 0, 2, 1, 2, 1, 0, 2, 1, 2, 2, 1, 2, 2, 2,
       1, 1, 1, 1, 0, 1, 1, 2, 2, 1], dtype=int64)
```

In [27]:
```
con_mat_test = confusion_matrix(Y_prediction_test, Y_test)
con_mat_test
```

Out[27]:
```
array([[10,  0,  0],
       [ 0,  9,  0],
       [ 0,  1, 10]], dtype=int64)
```

In [28]:
```
con_mat_train = confusion_matrix(Y_prediction_train, Y_train)
con_mat_train
```

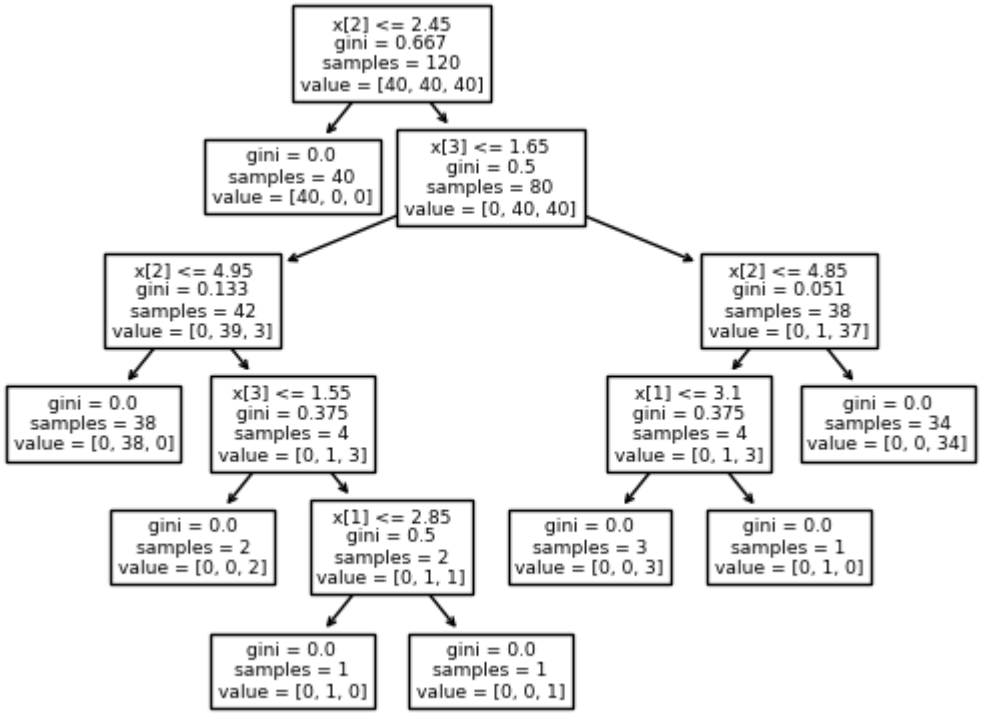
Out[28]:
```
array([[40,  0,  0],
       [ 0, 40,  0],
       [ 0,  0, 40]], dtype=int64)
```

In [29]:
```
print(classification_report(Y_prediction_train, Y_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       1.00      1.00      1.00        40
           2       1.00      1.00      1.00        40

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

In [30]:
```
plot_tree(dt_model.fit(X_train, Y_train))
```

Out[30]:
```
[Text(0.4, 0.9166666666666666, 'x[2] <= 2.45\ngini = 0.667\nsamples = 120\nvalue = [4
0, 40, 40]'),
 Text(0.3, 0.75, 'gini = 0.0\nsamples = 40\nvalue = [40, 0, 0]'),
 Text(0.5, 0.75, 'x[3] <= 1.65\ngini = 0.5\nsamples = 80\nvalue = [0, 40, 40]'),
 Text(0.2, 0.5833333333333334, 'x[2] <= 4.95\ngini = 0.133\nsamples = 42\nvalue = [0,
39, 3]'),
 Text(0.1, 0.4166666666666667, 'gini = 0.0\nsamples = 38\nvalue = [0, 38, 0]'),
 Text(0.3, 0.4166666666666667, 'x[3] <= 1.55\ngini = 0.375\nsamples = 4\nvalue = [0,
1, 3]'),
 Text(0.2, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.4, 0.25, 'x[1] <= 2.85\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]'),
 Text(0.3, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.5, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.8, 0.5833333333333334, 'x[2] <= 4.85\ngini = 0.051\nsamples = 38\nvalue = [0,
1, 37]'),
 Text(0.7, 0.4166666666666667, 'x[1] <= 3.1\ngini = 0.375\nsamples = 4\nvalue = [0,
1, 3]'),
 Text(0.6, 0.25, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]'),
 Text(0.8, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.9, 0.4166666666666667, 'gini = 0.0\nsamples = 34\nvalue = [0, 0, 34]')]
```


x[2] <= 2.45
gini = 0.667
samples = 120
value = [40, 40, 40]
gini = 0.0
samples = 40
value = [40, 0, 0]
x[3] <= 1.65
gini = 0.5
samples = 80
value = [0, 40, 40]
x[2] <= 4.95
gini = 0.133
samples = 42
value = [0, 39, 3]
x[2] <= 4.85
gini = 0.051
samples = 38
value = [0, 1, 37]
gini = 0.0
samples = 38
value = [0, 38, 0]
x[3] <= 1.55
gini = 0.375
samples = 4
value = [0, 1, 3]
x[1] <= 3.1
gini = 0.375
samples = 4
value = [0, 1, 3]
gini = 0.0
samples = 34
value = [0, 0, 34]
gini = 0.0
samples = 2
value = [0, 0, 2]
x[1] <= 2.85
gini = 0.5
samples = 2
value = [0, 1, 1]
gini = 0.0
samples = 3
value = [0, 0, 3]
gini = 0.0
samples = 1
value = [0, 1, 0]
gini = 0.0
samples = 1
value = [0, 1, 0]
gini = 0.0
samples = 1
value = [0, 0, 1]

In [ ]: