In [26]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
import matplotlib.pyplot as plt
```

In [14]:
```
import seaborn as sns
df=sns.load_dataset("iris")
df.head()
```

Out[14]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [15]:
```
df.columns
```

Out[15]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [18]:
```
## Separate the independent and dependent variables using the slicing metho
X=df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y=df[['species']]
```

In [20]:
```
## Split the data into training and testing sets.
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_sta
```

In [21]:
```
## Train the model using the decision tree classifier.
clf_gini=DecisionTreeClassifier(criterion="gini",random_state=100,max_depth
clf_gini.fit(X_train,y_train)
```

Out[21]:
```
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=100)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [22]: y_Predict_gini=clf_gini.predict(X_test)
         y_Predict_gini
```

```
Out[22]: array(['virginica', 'setosa', 'virginica', 'setosa', 'virginica',
                'virginica', 'setosa', 'setosa', 'virginica', 'setosa', 'setosa',
                'virginica', 'setosa', 'setosa', 'virginica', 'versicolor',
                'versicolor', 'virginica', 'virginica', 'virginica', 'virginica',
                'setosa', 'virginica', 'setosa', 'versicolor', 'virginica',
                'versicolor', 'setosa', 'versicolor', 'virginica', 'versicolor',
                'versicolor', 'versicolor', 'setosa', 'setosa', 'versicolor',
                'setosa', 'versicolor', 'virginica', 'virginica', 'setosa',
                'versicolor', 'virginica', 'virginica', 'setosa'], dtype=object)
```

```
In [24]: ## Calculate the accuracy of the model using the accuracy score function.
         print(("Accuracy is"),accuracy_score(y_test,y_Predict_gini)*100)
```

```
Accuracy is 95.55555555555556
```

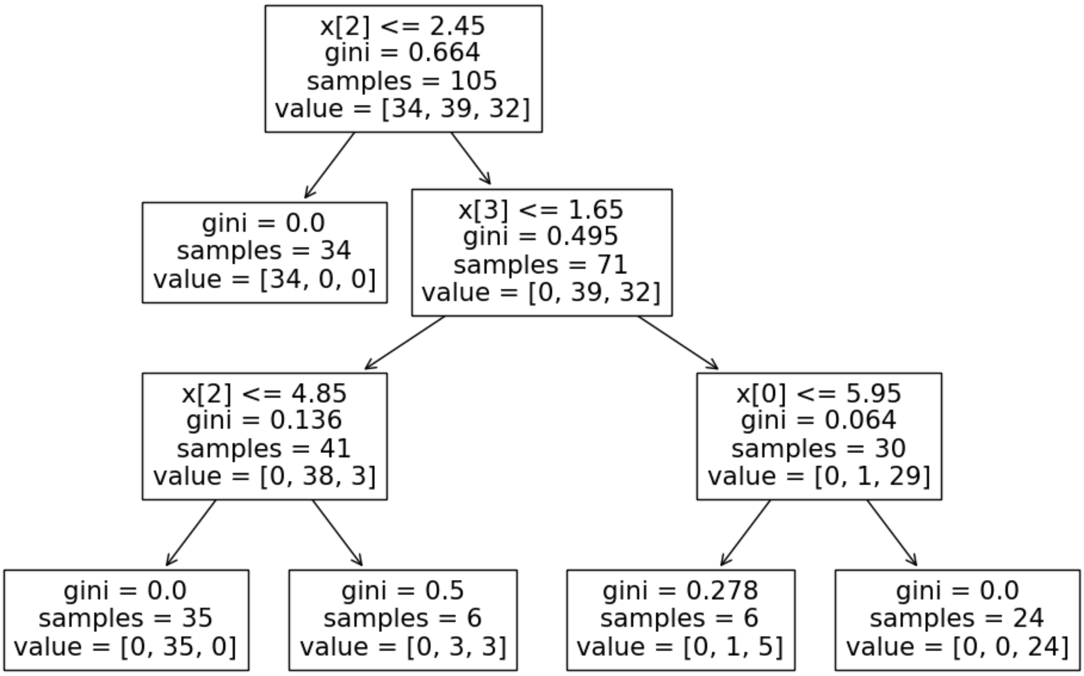
Our prediction model shows that there is an excellent accuracy score of 95.55555555555556 percent.

In [27]:

```
plt.figure(figsize=(12,8))
tree.plot_tree(clf_gini.fit(X_train, y_train))
```

Out[27]:

```
[Text(0.375, 0.875, 'x[2] <= 2.45\ngini = 0.664\nsamples = 105\nvalue = 
[34, 39, 32]'),
 Text(0.25, 0.625, 'gini = 0.0\nsamples = 34\nvalue = [34, 0, 0]'),
 Text(0.5, 0.625, 'x[3] <= 1.65\ngini = 0.495\nsamples = 71\nvalue = [0, 
39, 32]'),
 Text(0.25, 0.375, 'x[2] <= 4.85\ngini = 0.136\nsamples = 41\nvalue = [0, 
38, 3]'),
 Text(0.125, 0.125, 'gini = 0.0\nsamples = 35\nvalue = [0, 35, 0]'),
 Text(0.375, 0.125, 'gini = 0.5\nsamples = 6\nvalue = [0, 3, 3]'),
 Text(0.75, 0.375, 'x[0] <= 5.95\ngini = 0.064\nsamples = 30\nvalue = [0, 
1, 29]'),
 Text(0.625, 0.125, 'gini = 0.278\nsamples = 6\nvalue = [0, 1, 5]'),
 Text(0.875, 0.125, 'gini = 0.0\nsamples = 24\nvalue = [0, 0, 24]')]
```



In [28]:

```
## Entropy Method
## Train the model using the decision tree classifier.
clf_en=DecisionTreeClassifier(criterion="entropy",random_state=100,max_dept
clf_en.fit(X_train,y_train)
```

Out[28]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_leaf
=5,
                       random_state=100)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [29]:

```
y_Predict_en=clf_en.predict(X_test)
y_Predict_en
```

Out[29]:

```
array(['virginica', 'setosa', 'virginica', 'setosa', 'virginica',
       'virginica', 'setosa', 'setosa', 'virginica', 'setosa', 'setosa',
       'virginica', 'setosa', 'setosa', 'virginica', 'versicolor',
       'versicolor', 'virginica', 'virginica', 'virginica', 'virginica',
       'setosa', 'virginica', 'setosa', 'versicolor', 'virginica',
       'versicolor', 'setosa', 'versicolor', 'virginica', 'versicolor',
       'versicolor', 'versicolor', 'setosa', 'setosa', 'versicolor',
       'setosa', 'versicolor', 'virginica', 'virginica', 'setosa',
       'versicolor', 'virginica', 'virginica', 'setosa'], dtype=object)
```

In [30]:

```
## Calculate the accuracy of the model using the accuracy score function.
print(("Accuracy is"),accuracy_score(y_test,y_Predict_en)*100)
```

```
Accuracy is 95.55555555555556
```

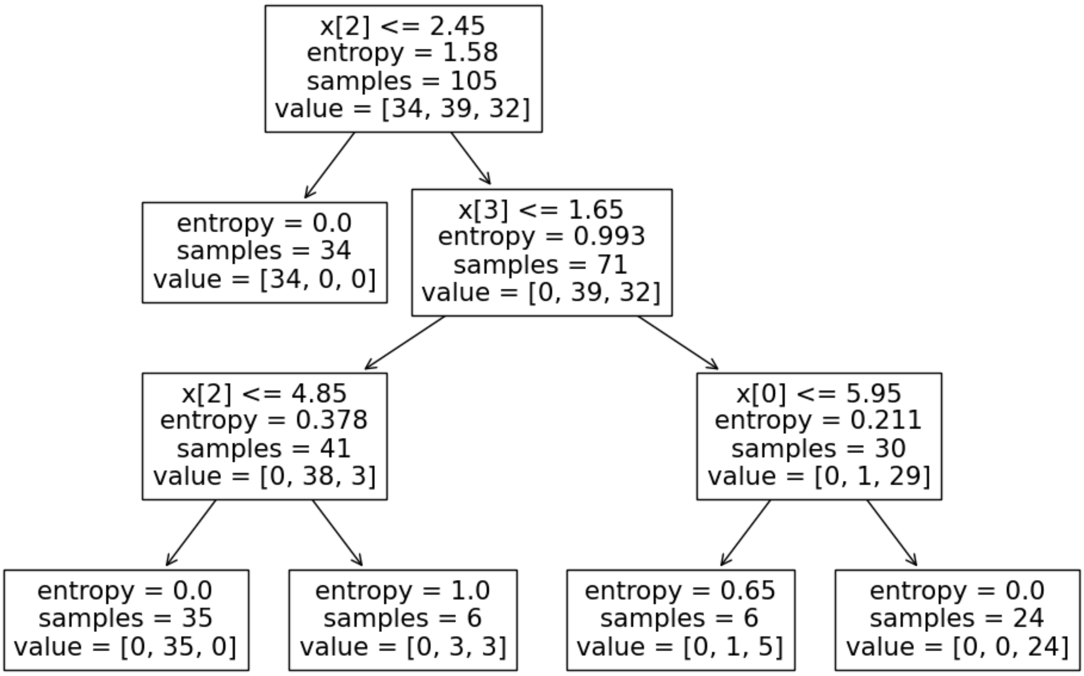
Visualize decision-trees

In [31]:

```
plt.figure(figsize=(12,8))
tree.plot_tree(clf_en.fit(X_train, y_train))
```

Out[31]:

```
[Text(0.375, 0.875, 'x[2] <= 2.45\nentropy = 1.58\nsamples = 105\nvalue =
[34, 39, 32]'),
 Text(0.25, 0.625, 'entropy = 0.0\nsamples = 34\nvalue = [34, 0, 0]'),
 Text(0.5, 0.625, 'x[3] <= 1.65\nentropy = 0.993\nsamples = 71\nvalue =
[0, 39, 32]'),
 Text(0.25, 0.375, 'x[2] <= 4.85\nentropy = 0.378\nsamples = 41\nvalue =
[0, 38, 3]'),
 Text(0.125, 0.125, 'entropy = 0.0\nsamples = 35\nvalue = [0, 35, 0]'),
 Text(0.375, 0.125, 'entropy = 1.0\nsamples = 6\nvalue = [0, 3, 3]'),
 Text(0.75, 0.375, 'x[0] <= 5.95\nentropy = 0.211\nsamples = 30\nvalue =
[0, 1, 29]'),
 Text(0.625, 0.125, 'entropy = 0.65\nsamples = 6\nvalue = [0, 1, 5]'),
 Text(0.875, 0.125, 'entropy = 0.0\nsamples = 24\nvalue = [0, 0, 24]')]
```



In [32]:
```
## Calculate the training set Accuracy
y_pred_train_en = clf_en.predict(X_train)
y_pred_train_en
```

Out[32]:
```
array(['setosa', 'setosa', 'setosa', 'versicolor', 'setosa', 'setosa',
       'setosa', 'versicolor', 'virginica', 'virginica', 'versicolor',
       'virginica', 'versicolor', 'virginica', 'virginica', 'versicolor',
       'setosa', 'virginica', 'virginica', 'versicolor', 'setosa',
       'setosa', 'virginica', 'setosa', 'setosa', 'setosa', 'versicolor',
       'virginica', 'virginica', 'versicolor', 'setosa', 'versicolor',
       'virginica', 'setosa', 'versicolor', 'versicolor', 'virginica',
       'setosa', 'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'virginica', 'virginica', 'setosa',
       'versicolor', 'setosa', 'versicolor', 'versicolor', 'virginica',
       'virginica', 'versicolor', 'setosa', 'setosa', 'setosa', 'setosa',
       'virginica', 'setosa', 'setosa', 'versicolor', 'setosa',
       'virginica', 'versicolor', 'virginica', 'setosa', 'virginica',
       'virginica', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'virginica', 'versicolor', 'versicolor', 'setosa', 'setosa',
       'setosa', 'virginica', 'versicolor', 'versicolor', 'setosa',
       'virginica', 'virginica', 'virginica', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'virginica', 'virginica', 'setosa',
       'setosa', 'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'virginica', 'versicolor', 'versicolor', 'virginica', 'versicolo
r',
       'setosa', 'setosa'], dtype=object)
```

In [33]:
```
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_tra
```

```
Training-set accuracy score: 0.9619
```

In [34]:
```
## Check for Overfitting and Underfitting
## print the scores on training and test set

print('Training set score: {:.4f}'.format(clf_en.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(clf_en.score(X_test, y_test)))
```

```
Training set score: 0.9619
Test set score: 0.9556
```

In [38]:
```
# Print the Confusion Matrix and slice it into four pieces

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_Predict_en)

print('Confusion matrix\n\n', cm)
```

```
Confusion matrix

 [[16  0  0]
 [ 0 10  1]
 [ 0  1 17]]
```

We can see that the training-set score and test-set score is same as above. The training-set accuracy score is 0.9619 while the test-set accuracy to be 0.9556. These

two values are quite comparable. So, there is no sign of overfitting.

```
In [ ]:
```