In [4]:
```
#Aim: To implement decision tree algorithm for iris dataset
```

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import classification_report
from sklearn import metrics
import seaborn as sns
```

In [6]:
```
dataset=pd.read_csv("Logistic_Iris.csv")
dataset
```

Out[6]:

	Sepal Length	Sepal Width	Petal Length	Peatal Width	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [7]:
```
x=dataset.iloc[:,[0,1,2,3]]
x
```

Out[7]:

	Sepal Length	Sepal Width	Petal Length	Peatal Width
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
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [8]:
```
y=dataset.iloc[:,4]
y
```

Out[8]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

In [9]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=42)
```

In [10]:
```
xtrain
```

Out[10]:

	Sepal Length	Sepal Width	Petal Length	Peatal Width
4	5.0	3.6	1.4	0.2
32	5.2	4.1	1.5	0.1
142	5.8	2.7	5.1	1.9
85	6.0	3.4	4.5	1.6
86	6.7	3.1	4.7	1.5
...	...	...	...	...
71	6.1	2.8	4.0	1.3
106	4.9	2.5	4.5	1.7
14	5.8	4.0	1.2	0.2
92	5.8	2.6	4.0	1.2
102	7.1	3.0	5.9	2.1

112 rows × 4 columns

In [11]:
```
sc=StandardScaler()
xtrain=sc.fit_transform(xtrain)
xtest=sc.fit_transform(xtest)
```

In [12]:
```
dtree_gini=DecisionTreeClassifier(criterion="gini",random_state=100,max_depth=3,min_samples_leaf=4)
dtree_gini.fit(xtrain,ytrain)
```

Out[12]:
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, min_samples_leaf=4, random_state=100)
```

In [13]:
```
DecisionTreeClassifier(max_depth=3,min_samples_leaf=4,random_state=100)
```

Out[13]:
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, min_samples_leaf=4, random_state=100)
```

In [14]:
```
y_pred1=dtree_gini.predict(xtest)
```

In [15]:
```
accgini=accuracy_score(ytest,y_pred1)*100
print("\n\nAccuracy using gini index:",accgini)
```

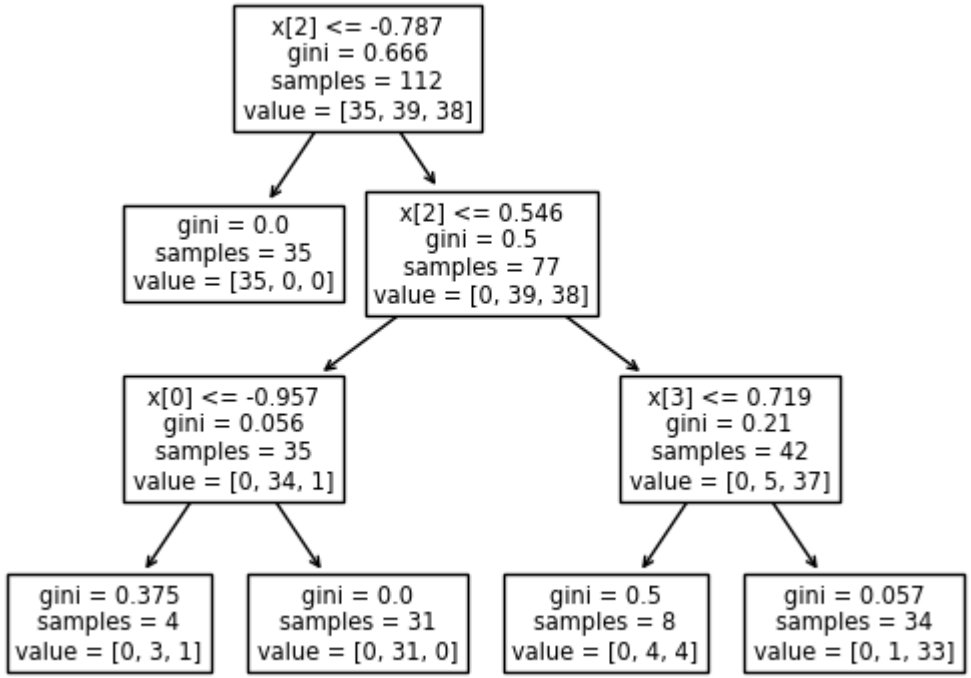
```
Accuracy using gini index: 100.0
```

In [16]:
```
cm=confusion_matrix(ytest,y_pred1)
print("confusion matrix:\n",cm)
```

```
confusion matrix:
 [[15  0  0]
 [ 0 11  0]
 [ 0  0 12]]
```

In [17]:
```
tree.plot_tree(dtree_gini)
```

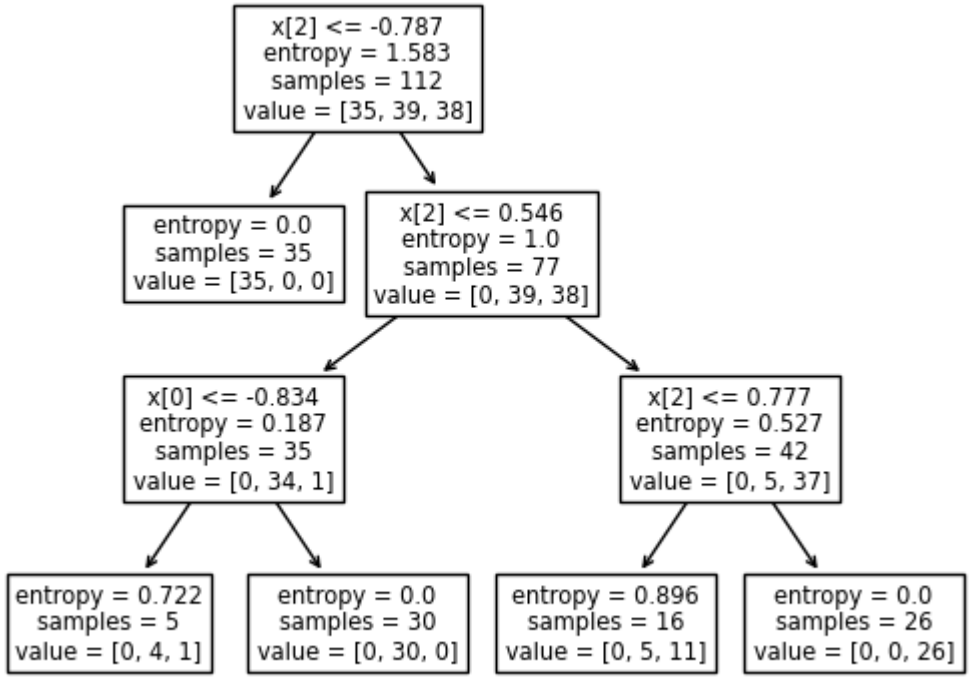
Out[17]:
```
[Text(0.375, 0.875, 'x[2] <= -0.787\ngini = 0.666\nsamples = 112\nvalue = [35, 39, 38]'),
 Text(0.25, 0.625, 'gini = 0.0\nsamples = 35\nvalue = [35, 0, 0]'),
 Text(0.5, 0.625, 'x[2] <= 0.546\ngini = 0.5\nsamples = 77\nvalue = [0, 39, 38]'),
 Text(0.25, 0.375, 'x[0] <= -0.957\ngini = 0.056\nsamples = 35\nvalue = [0, 34, 1]'),
 Text(0.125, 0.125, 'gini = 0.375\nsamples = 4\nvalue = [0, 3, 1]'),
 Text(0.375, 0.125, 'gini = 0.0\nsamples = 31\nvalue = [0, 31, 0]'),
 Text(0.75, 0.375, 'x[3] <= 0.719\ngini = 0.21\nsamples = 42\nvalue = [0, 5, 37]'),
 Text(0.625, 0.125, 'gini = 0.5\nsamples = 8\nvalue = [0, 4, 4]'),
 Text(0.875, 0.125, 'gini = 0.057\nsamples = 34\nvalue = [0, 1, 33]')]
```



In [18]:
```
dtree_entropy=DecisionTreeClassifier(criterion="entropy",random_state=100,max_depth=3,min_samples_leaf=5)
dtree_entropy.fit(xtrain,ytrain)
y_pred3=dtree_entropy.predict(xtest)
```

In [19]:
```
tree.plot_tree(dtree_entropy)
```

Out[19]:
```
[Text(0.375, 0.875, 'x[2] <= -0.787\nentropy = 1.583\nsamples = 112\nvalue = [35, 39, 38]'),
 Text(0.25, 0.625, 'entropy = 0.0\nsamples = 35\nvalue = [35, 0, 0]'),
 Text(0.5, 0.625, 'x[2] <= 0.546\nentropy = 1.0\nsamples = 77\nvalue = [0, 39, 38]'),
 Text(0.25, 0.375, 'x[0] <= -0.834\nentropy = 0.187\nsamples = 35\nvalue = [0, 34, 1]'),
 Text(0.125, 0.125, 'entropy = 0.722\nsamples = 5\nvalue = [0, 4, 1]'),
 Text(0.375, 0.125, 'entropy = 0.0\nsamples = 30\nvalue = [0, 30, 0]'),
 Text(0.75, 0.375, 'x[2] <= 0.777\nentropy = 0.527\nsamples = 42\nvalue = [0, 5, 37]'),
 Text(0.625, 0.125, 'entropy = 0.896\nsamples = 16\nvalue = [0, 5, 11]'),
 Text(0.875, 0.125, 'entropy = 0.0\nsamples = 26\nvalue = [0, 0, 26]')]
```



In [20]:
```
accentropy=accuracy_score(ytest,y_pred3)*100
print("\n\nAccuracy using gini index:",accentropy)
```

```
Accuracy using gini index: 92.10526315789474
```

In [21]:
```
cm2=confusion_matrix(ytest,y_pred3)
print("confusion matrix:\n",cm2)
```

```
confusion matrix:
 [[15  0  0]
 [ 0  8  3]
 [ 0  0 12]]
```

In [22]:
```
fig,ax=plt.subplots(figsize=(6,6))
ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0,1,2),ticklabels=("predicted setosa","predicted Versicolor","predicted Virginica"))
ax.yaxis.set(ticks=(0,1,2),ticklabels=("Actual Setosa","Actual Versicolor","Actual Virginica"))
ax.set_ylim(2.5,-0.5)
for i in range(3):
    for j in range(3):
        ax.text(j,i,cm[i,j],ha="center",va="center",color="white")
plt.show()
```