In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [7]:
```
from sklearn import datasets

iris_data = datasets.load_iris()
```

In [8]:
```
1 print(iris_data.DESCR)
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to
a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysi
```

```
s.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactio
ns
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

In [10]:
```
inp = iris_data['data']
out = iris_data['target']
```

In [13]:
```
col_names = iris_data['feature_names']
col_names
```

Out[13]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [17]:
```
inp_df = pd.DataFrame(inp, columns = col_names)
out_df = pd.DataFrame(out, columns = ['target'])
```

In [19]:
```
iris = pd.concat([inp_df, out_df], axis = 1)
iris
```

Out[19]:

	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

In [29]:
```
iris.target = iris.target.map({0:'setosa',1:'versicolor',2:'verginica'})
```

In [30]:
```
1 iris.target.value_counts()
```

Out[30]:
```
setosa        50
versicolor    50
verginica     50
Name: target, dtype: int64
```

Train-Test split

In [33]:
```
1 from sklearn.model_selection import train_test_split
```

In [58]:
```
1 X = iris.drop('target', axis = 1)
2 y = iris['target']
3
4 X_train, X_test, y_train, y_test = train_test_split(X,y, random_state = 1, t
```

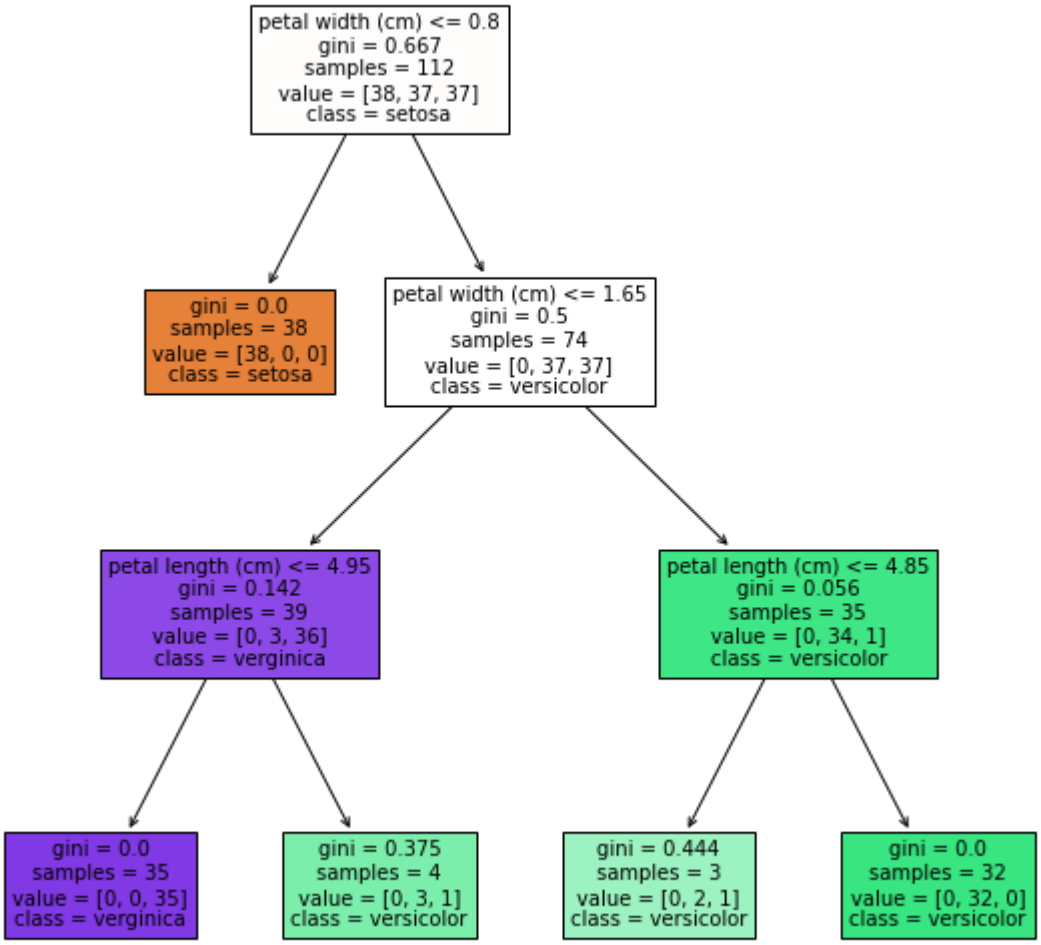
Model Building

In [59]:
```
1 from sklearn.tree import DecisionTreeClassifier, plot_tree
```

In [94]:
```
1 dt_model = DecisionTreeClassifier(criterion = 'gini',max_depth = 3)
2 dt_model.fit(X_train, y_train)
```

Out[94]:
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

In [95]:
```
plt.figure(figsize = (10,10))
plot_tree(dt_model,feature_names=col_names,class_names=['setosa','versicolor
plt.show()
```



In [96]:
```
plot_tree?
```

Prediction

In [97]:
```
y_pred = dt_model.predict(X_test)
y_pred
```

Out[97]:
```
array(['versicolor', 'setosa', 'setosa', 'setosa', 'verginica', 'setosa',
       'verginica', 'versicolor', 'setosa', 'verginica', 'versicolor',
       'verginica', 'versicolor', 'verginica', 'versicolor', 'verginica',
       'versicolor', 'verginica', 'verginica', 'verginica', 'verginica',
       'versicolor', 'versicolor', 'verginica', 'setosa', 'setosa',
       'setosa', 'verginica', 'versicolor', 'setosa', 'setosa',
       'versicolor', 'verginica', 'setosa', 'setosa', 'verginica',
       'versicolor', 'versicolor'], dtype=object)
```

Evaluation

Accuracy Score

In [98]:
```
dt_model.score(X_test,y_test)
```

Out[98]:
```
0.9736842105263158
```

precision, recall, confusion_matrix, classification report

In [91]:
```
from sklearn.metrics import precision_score, recall_score, confusion_matrix,
```

In [92]:
```
# precsion score

precision_score(y_test, y_pred, average = 'macro')
```

Out[92]:
```
0.9761904761904763
```

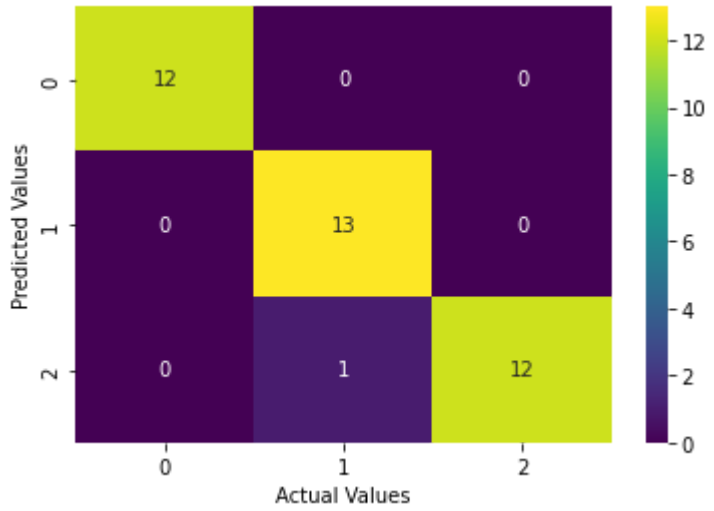
In [93]:
```
# recall score

recall_score(y_test, y_pred, average = 'macro')
```

Out[93]:
```
0.9743589743589745
```

Confusion Matrix

In [85]:
```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot = True, cmap = 'viridis')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.show()
```



Classification Report

In [86]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        12
   verginica       0.93      1.00      0.96        13
  versicolor       1.00      0.92      0.96        13

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

In [ ]:
```

```