In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MaxAbsScaler
from sklearn.model_selection import train_test_split
from collections import Counter
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [2]:

```
df=sns.load_dataset("iris")
```

In [3]:

```
df.head()
```

Out[3]:

	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 [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:

```
l1=LabelEncoder()
df["species"]=l1.fit_transform(df["species"])
```

In [6]:

```
df.head()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width	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 [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    int32  
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

In [8]:

```
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

In [9]:

```
mas=MaxAbsScaler()
X=mas.fit_transform(x)
```

In [10]:

```
X=pd.DataFrame(X,columns=["Sepal_l","Sepal_w","Petal_l","Petal_w"])
```

In [11]:

```
X
```

Out[11]:

	Sepal_l	Sepal_w	Petal_l	Petal_w
0	0.645570	0.795455	0.202899	0.08
1	0.620253	0.681818	0.202899	0.08
2	0.594937	0.727273	0.188406	0.08
3	0.582278	0.704545	0.217391	0.08
4	0.632911	0.818182	0.202899	0.08
...	...	...	...	...
145	0.848101	0.681818	0.753623	0.92
146	0.797468	0.568182	0.724638	0.76
147	0.822785	0.681818	0.753623	0.80
148	0.784810	0.772727	0.782609	0.92

In [12]:

```
x_train,x_test,y_train,y_test=train_test_split(X,y,random_state=1)
print(Counter(y_test))
```

```
Counter({1: 16, 0: 13, 2: 9})
```

In [13]:

```
dt=DecisionTreeClassifier(max_depth=3,random_state=1)
```

In [14]:

```
d=dt.fit(x_train,y_train)
```

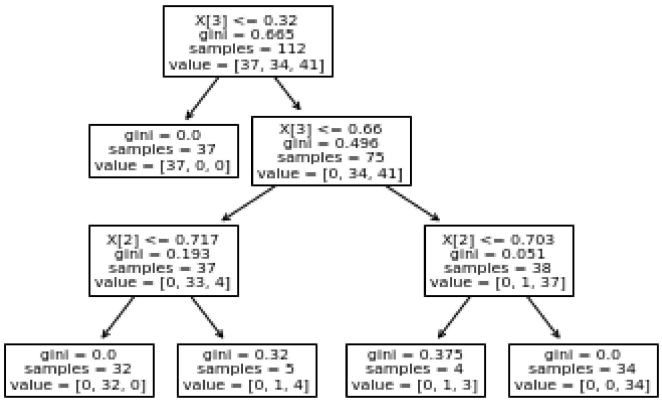
plotting tree

In [15]:

```
tree.plot_tree(d)
```

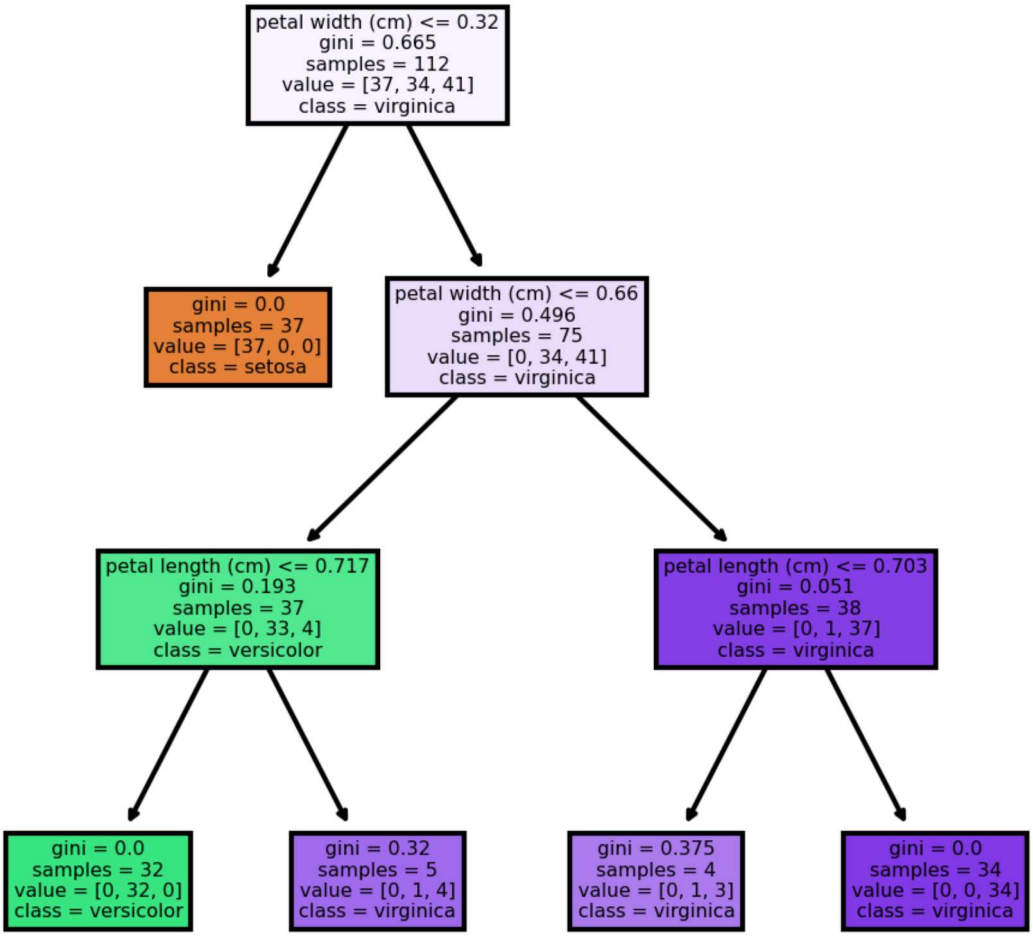
Out[15]:

```
[Text(0.375, 0.875, 'X[3] <= 0.32\ngini = 0.665\nsamples = 112\nvalue = [37,
34, 41]'),
 Text(0.25, 0.625, 'gini = 0.0\nsamples = 37\nvalue = [37, 0, 0]'),
 Text(0.5, 0.625, 'X[3] <= 0.66\ngini = 0.496\nsamples = 75\nvalue = [0, 34,
41]'),
 Text(0.25, 0.375, 'X[2] <= 0.717\ngini = 0.193\nsamples = 37\nvalue = [0, 3
3, 4]'),
 Text(0.125, 0.125, 'gini = 0.0\nsamples = 32\nvalue = [0, 32, 0]'),
 Text(0.375, 0.125, 'gini = 0.32\nsamples = 5\nvalue = [0, 1, 4]'),
 Text(0.75, 0.375, 'X[2] <= 0.703\ngini = 0.051\nsamples = 38\nvalue = [0,
1, 37]'),
 Text(0.625, 0.125, 'gini = 0.375\nsamples = 4\nvalue = [0, 1, 3]'),
 Text(0.875, 0.125, 'gini = 0.0\nsamples = 34\nvalue = [0, 0, 34]')]
```



In [16]:

```
f_n=['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)']
c_n=['setosa', 'versicolor', 'virginica']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
tree.plot_tree(d,
               feature_names = f_n,
               class_names=c_n,
               filled = True);
fig.savefig('imagename.png')
```



prediction

In [17]:

```
d_pred=dt.predict(x_test)
```

In [18]:

```
print(d_pred)
```

```
[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]
```

Checking Performance

In [19]:

```
d_cm=confusion_matrix(y_test,d_pred)
d_cm
```

Out[19]:

```
array([[13,  0,  0],
       [ 0, 15,  1],
       [ 0,  0,  9]], dtype=int64)
```

In [20]:

```
d_ac=accuracy_score(y_test,d_pred)
print((d_ac.round(2)))
```

```
0.97
```

In [21]:

```
d_cr=classification_report(y_test,d_pred)
print(d_cr)
```

```
              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
```