In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import  DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import classification_report
from sklearn import preprocessing
```

In [2]:

```
# import some data to play with
iris = pd.read_csv('C:/Users/Ashraf/Documents/Datafiles/iris.csv',index_col=0)
iris.head()
```

Out[2]:

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

In [3]:

```
#Complete Iris dataset
label_encoder = preprocessing.LabelEncoder()
iris['Species']= label_encoder.fit_transform(iris['Species'])
```

In [4]:

```
x=iris.iloc[:,0:4]
y=iris['Species']
```

In [5]:

```
x
```

Out[5]:

| Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|
| **1** | 5.1 | 3.5 | 1.4 | 0.2 |
| **2** | 4.9 | 3.0 | 1.4 | 0.2 |
| **3** | 4.7 | 3.2 | 1.3 | 0.2 |
| **4** | 4.6 | 3.1 | 1.5 | 0.2 |
| **5** | 5.0 | 3.6 | 1.4 | 0.2 |
| **...** | ... | ... | ... | ... |
| **146** | 6.7 | 3.0 | 5.2 | 2.3 |
| **147** | 6.3 | 2.5 | 5.0 | 1.9 |
| **148** | 6.5 | 3.0 | 5.2 | 2.0 |
| **149** | 6.2 | 3.4 | 5.4 | 2.3 |
| **150** | 5.9 | 3.0 | 5.1 | 1.8 |

150 rows × 4 columns

In [6]:

```
y
```

Out[6]:

```
Id
1      0
2      0
3      0
4      0
5      0
      ..
146    2
147    2
148    2
149    2
150    2
Name: Species, Length: 150, dtype: int32
```

In [7]:

```
iris['Species'].unique()
```

Out[7]:

```
array([0, 1, 2])
```

In [8]:

```
iris.Species.value_counts()
```

Out[8]:

```
0    50
1    50
2    50
Name: Species, dtype: int64
```

In [9]:

```
colnames = list(iris.columns)
colnames
```

Out[9]:

```
['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'Specie
s']
```

In [10]:

```
# Splitting data into training and testing data set
x_train, x_test,y_train,y_test = train_test_split(x,y, test_size=0.2,random_state=40)
```

# Building Decision Tree Classifier using Entropy Criteria

In [11]:

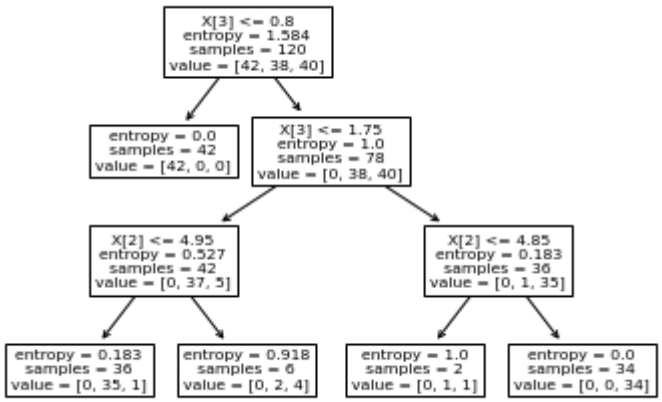
```
model = DecisionTreeClassifier(criterion = 'entropy',max_depth=3)
model.fit(x_train,y_train)
```

Out[11]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

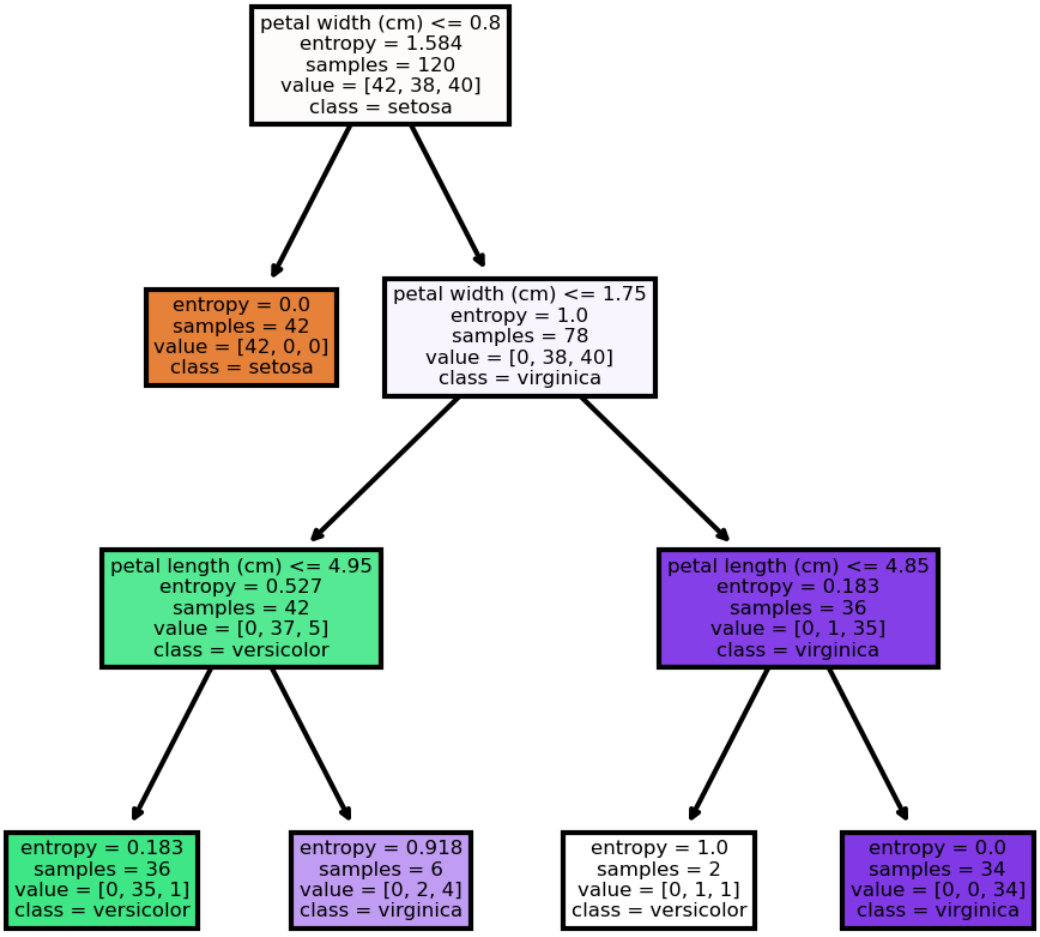
In [12]:

```
#PLot the decision tree
tree.plot_tree(model);
```



In [13]:

```
fn=['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)']
cn=['setosa', 'versicolor', 'virginica']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
tree.plot_tree(model,
               feature_names = fn,
               class_names=cn,
               filled = True);
```



In [14]:

```
#Predicting on test data
preds = model.predict(x_test) # predicting on test data set
pd.Series(preds).value_counts() # getting the count of each cate
```

Out[14]:

```
1    13
2     9
0     8
dtype: int64
```

In [15]:

```
preds
```

Out[15]:

```
array([0, 1, 2, 2, 1, 2, 1, 1, 1, 0, 1, 0, 0, 1, 1, 2, 2, 2, 1, 1, 2, 2,
       1, 0, 1, 0, 0, 2, 0, 1])
```

In [16]:

```
pd.crosstab(y_test,preds) # getting the 2 way table to understand the correct and wrong pre
```

Out[16]:

| col_0 | 0 | 1 | 2 |
|---|---|---|---|
| Species | | | |
| **0** | 8 | 0 | 0 |
| **1** | 0 | 12 | 0 |
| **2** | 0 | 1 | 9 |

In [17]:

```
# Accuracy
np.mean(preds==y_test)
```

Out[17]:

```
0.9666666666666667
```

# Building Decision Tree Classifier (CART) using Gini Criteria

In [18]:

```
# Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor
```

In [19]:

```
array = iris.values
X = array[:,0:3]
y = array[:,3]
```

In [20]:

```
x
```

Out[20]:

| Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|
| **1** | 5.1 | 3.5 | 1.4 | 0.2 |
| **2** | 4.9 | 3.0 | 1.4 | 0.2 |
| **3** | 4.7 | 3.2 | 1.3 | 0.2 |
| **4** | 4.6 | 3.1 | 1.5 | 0.2 |
| **5** | 5.0 | 3.6 | 1.4 | 0.2 |
| **...** | ... | ... | ... | ... |
| **146** | 6.7 | 3.0 | 5.2 | 2.3 |
| **147** | 6.3 | 2.5 | 5.0 | 1.9 |
| **148** | 6.5 | 3.0 | 5.2 | 2.0 |
| **149** | 6.2 | 3.4 | 5.4 | 2.3 |
| **150** | 5.9 | 3.0 | 5.1 | 1.8 |

150 rows × 4 columns

In [21]:

```
y
```

Out[21]:

```
array([0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.2, 0.1,
       0.1, 0.2, 0.4, 0.4, 0.3, 0.3, 0.3, 0.2, 0.4, 0.2, 0.5, 0.2, 0.2,
       0.4, 0.2, 0.2, 0.2, 0.2, 0.4, 0.1, 0.2, 0.1, 0.2, 0.2, 0.1, 0.2,
       0.2, 0.3, 0.3, 0.2, 0.6, 0.4, 0.3, 0.2, 0.2, 0.2, 0.2, 1.4, 1.5,
       1.5, 1.3, 1.5, 1.3, 1.6, 1. , 1.3, 1.4, 1. , 1.5, 1. , 1.4, 1.3,
       1.4, 1.5, 1. , 1.5, 1.1, 1.8, 1.3, 1.5, 1.2, 1.3, 1.4, 1.4, 1.7,
       1.5, 1. , 1.1, 1. , 1.2, 1.6, 1.5, 1.6, 1.5, 1.3, 1.3, 1.3, 1.2,
       1.4, 1.2, 1. , 1.3, 1.2, 1.3, 1.3, 1.1, 1.3, 2.5, 1.9, 2.1, 1.8,
       2.2, 2.1, 1.7, 1.8, 1.8, 2.5, 2. , 1.9, 2.1, 2. , 2.4, 2.3, 1.8,
       2.2, 2.3, 1.5, 2.3, 2. , 2. , 1.8, 2.1, 1.8, 1.8, 1.8, 2.1, 1.6,
       1.9, 2. , 2.2, 1.5, 1.4, 2.3, 2.4, 1.8, 1.8, 2.1, 2.4, 2.3, 1.9,
       2.3, 2.5, 2.3, 1.9, 2. , 2.3, 1.8])
```

In [22]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
```

In [23]:

```
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
```

Out[23]:

```
DecisionTreeRegressor()
```

In [24]:

```
#Find the accuracy
model.score(X_test,y_test)
```

Out[24]:

```
0.8559250810421419
```

In [ ]:

```

```