Charger la librairie Pandas

In [2]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Etape 1 : Préparation des données

1- Téléchargement des données

In [5]:
```
df = pd.read_csv('iris.csv')
```

Afficher les six premières lignes du jeu de données

In [7]:
```
print(df.head(10))
```

```
   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
5           5.4          3.9           1.7          0.4  setosa
6           4.6          3.4           1.4          0.3  setosa
7           5.0          3.4           1.5          0.2  setosa
8           4.4          2.9           1.4          0.2  setosa
9           4.9          3.1           1.5          0.1  setosa
```

In [8]:
```
display(df)
```

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

2-Détermination du nombre d'individus et de variables

Dimensions de la table de données : nombre de lignes, nombre de colonnes
la ligne d'en-tête n'est pas comptabilisée dans le nombre de lignes

In [10]:
```
dimension = df.shape
NbrLignes = df.shape[0]
NbrColonnes = df.shape[1]
```

In [11]:
```
print("Dimension :",dimension)
print("Nombre de lignes :",NbrLignes)
print("Nombre de colonnes :",NbrColonnes)
```

```
Dimension : (150, 5)
Nombre de lignes : 150
Nombre de colonnes : 5
```

Énumération des variables

In [13]:
```
print(df.columns)
print(df.dtypes)
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

La base de données contient 150 individus = 150 échantillons. Le nombre de caractéristique est 4: sepal_length, sepal_width, petal_length, petal_width. Les quatres caractéristiques sont de nature quantitatives (float64)

In [15]:
```
Modalites=df['species'].unique()
print('Les modalités sont: ',Modalites)
```

```
Les modalités sont:  ['setosa' 'versicolor' 'virginica']
```

Les modalités sont: 'setosa' 'versicolor' 'virginica'

In [17]:
```
E1 = df['species'].value_counts()
print('Effectif :\n',E1)
```

```
Effectif :
 species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

3- Répartissez les données en deux ensembles égaux

In [19]:
```
train, test = train_test_split(df, test_size = 0.5, random_state = 10)
train.groupby('species').size()
```

Out[19]:
```
species
setosa        27
versicolor    23
virginica     25
dtype: int64
```

In [20]:
```
### 5- Répartissez les données en deux ensembles égaux
```

In [21]:
```
train, test = train_test_split(df, test_size = 0.33, random_state = 10)
train.groupby('species').size()
```

Out[21]:
```
species
setosa        35
versicolor    31
virginica     34
dtype: int64
```

6- Créer deux matrices contenant les vecteurs de caractéristiques

In [23]:
```
X_train = train[['sepal_length','sepal_width','petal_length','petal_width']]
y_train = train.species
X_test = test[['sepal_length','sepal_width','petal_length','petal_width']]
y_test = test.species
fn = ['sepal_length', "sepal_width", "petal_length", "petal_width"]
cn = ['setosa', 'versicolor', 'virginica']
```

Etape2 : Classification

In [25]:
```
from sklearn import tree
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

1- Construction d'un arbre de décision (hauteur =3, random_state = 1)

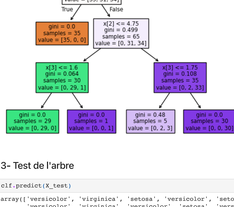
In [27]:
```
clf = tree.DecisionTreeClassifier(max_depth = 3)
clf.fit(X_train, y_train)
```

Out[27]:
```
DecisionTreeClassifier(max_depth=3)
```

2 - Visualisation de l'arbre de décision

In [29]:
```
tree.plot_tree(clf, filled=True)
```

Out[29]:
```
[Text(0.375, 0.875, 'x[2] <= 2.45\ngini = 0.666\nsamples = 100\nvalue = [35, 31, 34]'),
 Text(0.25, 0.625, 'gini = 0.0\nsamples = 35\nvalue = [35, 0, 0]'),
 Text(0.3125, 0.75, 'True  '),
 Text(0.5, 0.625, 'x[2] <= 4.75\ngini = 0.499\nsamples = 65\nvalue = [0, 31, 34]'),
 Text(0.4375, 0.75, '  False'),
 Text(0.25, 0.375, 'x[3] <= 1.6\ngini = 0.064\nsamples = 30\nvalue = [0, 29, 1]'),
 Text(0.125, 0.125, 'gini = 0.0\nsamples = 29\nvalue = [0, 29, 0]'),
 Text(0.375, 0.125, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.75, 0.375, 'x[3] <= 1.75\ngini = 0.108\nsamples = 35\nvalue = [0, 2, 33]'),
 Text(0.625, 0.125, 'gini = 0.48\nsamples = 5\nvalue = [0, 2, 3]'),
 Text(0.875, 0.125, 'gini = 0.0\nsamples = 30\nvalue = [0, 0, 30]')]
```



3- Test de l'arbre

In [31]:
```
clf.predict(X_test)
```

Out[31]:
```
array(['versicolor', 'virginica', 'setosa', 'versicolor', 'setosa',
       'versicolor', 'virginica', 'versicolor', 'setosa', 'versicolor',
       'versicolor', 'virginica', 'versicolor', 'setosa', 'setosa',
       'virginica', 'versicolor', 'setosa', 'setosa', 'setosa',
       'virginica', 'virginica', 'virginica', 'setosa', 'versicolor',
       'setosa', 'versicolor', 'versicolor', 'virginica', 'virginica',
       'versicolor', 'versicolor', 'virginica', 'virginica', 'virginica',
       'setosa', 'virginica', 'virginica', 'virginica', 'virginica',
       'setosa', 'setosa', 'versicolor', 'setosa', 'versicolor', 'setosa',
       'virginica', 'virginica', 'virginica', 'virginica'], dtype=object)
```

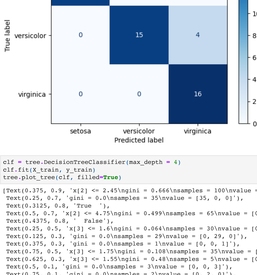
In [32]:
```
clf.score(X_test, y_test)
```

Out[32]:
```
0.92
```

4- Matrice de confusion

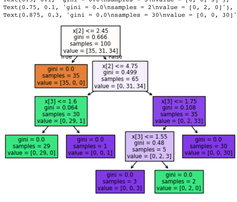
In [34]:
```
disp = ConfusionMatrixDisplay.from_estimator(
        clf,
        X_test,
        y_test,
        display_labels=cn,
        cmap=plt.cm.Blues,
    )

print(disp.confusion_matrix)
plt.show()
```

```
[[15  0  0]
 [ 0 15  4]
 [ 0  0 16]]
```



In [35]:
```
clf = tree.DecisionTreeClassifier(max_depth = 4)
clf.fit(X_train, y_train)
tree.plot_tree(clf, filled=True)
```

Out[35]:
```
[Text(0.375, 0.9, 'x[2] <= 2.45\ngini = 0.666\nsamples = 100\nvalue = [35, 31, 34]'),
 Text(0.25, 0.7, 'gini = 0.0\nsamples = 35\nvalue = [35, 0, 0]'),
 Text(0.3125, 0.8, 'True  '),
 Text(0.5, 0.7, 'x[2] <= 4.75\ngini = 0.499\nsamples = 65\nvalue = [0, 31, 34]'),
 Text(0.4375, 0.8,   'False'),
 Text(0.25, 0.5, 'x[3] <= 1.6\ngini = 0.064\nsamples = 30\nvalue = [0, 29, 1]'),
 Text(0.125, 0.3, 'gini = 0.0\nsamples = 29\nvalue = [0, 29, 0]'),
 Text(0.375, 0.3, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.75, 0.5, 'x[3] <= 1.75\ngini = 0.108\nsamples = 35\nvalue = [0, 2, 33]'),
 Text(0.625, 0.3, 'x[3] <= 1.55\ngini = 0.48\nsamples = 5\nvalue = [0, 2, 3]'),
 Text(0.5, 0.1, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]'),
 Text(0.75, 0.1, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]'),
 Text(0.875, 0.3, 'gini = 0.0\nsamples = 30\nvalue = [0, 0, 30]')]
```



In [36]:
```
clf.predict(X_test)
```

Out[36]:
```
array(['versicolor', 'virginica', 'setosa', 'versicolor', 'setosa',
       'versicolor', 'virginica', 'versicolor', 'setosa', 'versicolor',
       'versicolor', 'virginica', 'versicolor', 'setosa', 'setosa',
       'virginica', 'versicolor', 'setosa', 'setosa', 'setosa',
       'virginica', 'virginica', 'virginica', 'setosa', 'versicolor',
       'setosa', 'versicolor', 'versicolor', 'virginica', 'virginica',
       'versicolor', 'versicolor', 'virginica', 'virginica', 'virginica',
       'setosa', 'virginica', 'virginica', 'virginica', 'virginica',
       'setosa', 'setosa', 'versicolor', 'setosa', 'versicolor', 'setosa',
       'virginica', 'virginica', 'virginica', 'virginica'], dtype=object)
```

In [37]:
```
clf.score(X_test, y_test)
```

Out[37]:
```
0.9
```

In [ ]:
```

```