Importation des bibliothèques

In [1]:

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
import matplotlib.pyplot as plt
```

Chargement du dataset Iris

In [2]:

```
iris = datasets.load_iris()
X = iris.data  # Features (sepal length, sepal width, petal length, petal width)
y = iris.target  # Labels
```

Division en jeu d'entraînement et test

In [3]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42
)
```

Entraînement du Decision Tree

In [4]:

```
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
```

Out[4]:

▾ DecisionTreeClassifier i ?

DecisionTreeClassifier(random_state=42)

Prédiction sur les données test

In [5]:

```
y_pred = clf.predict(X_test)
```

Évaluation de la performance

In [6]:

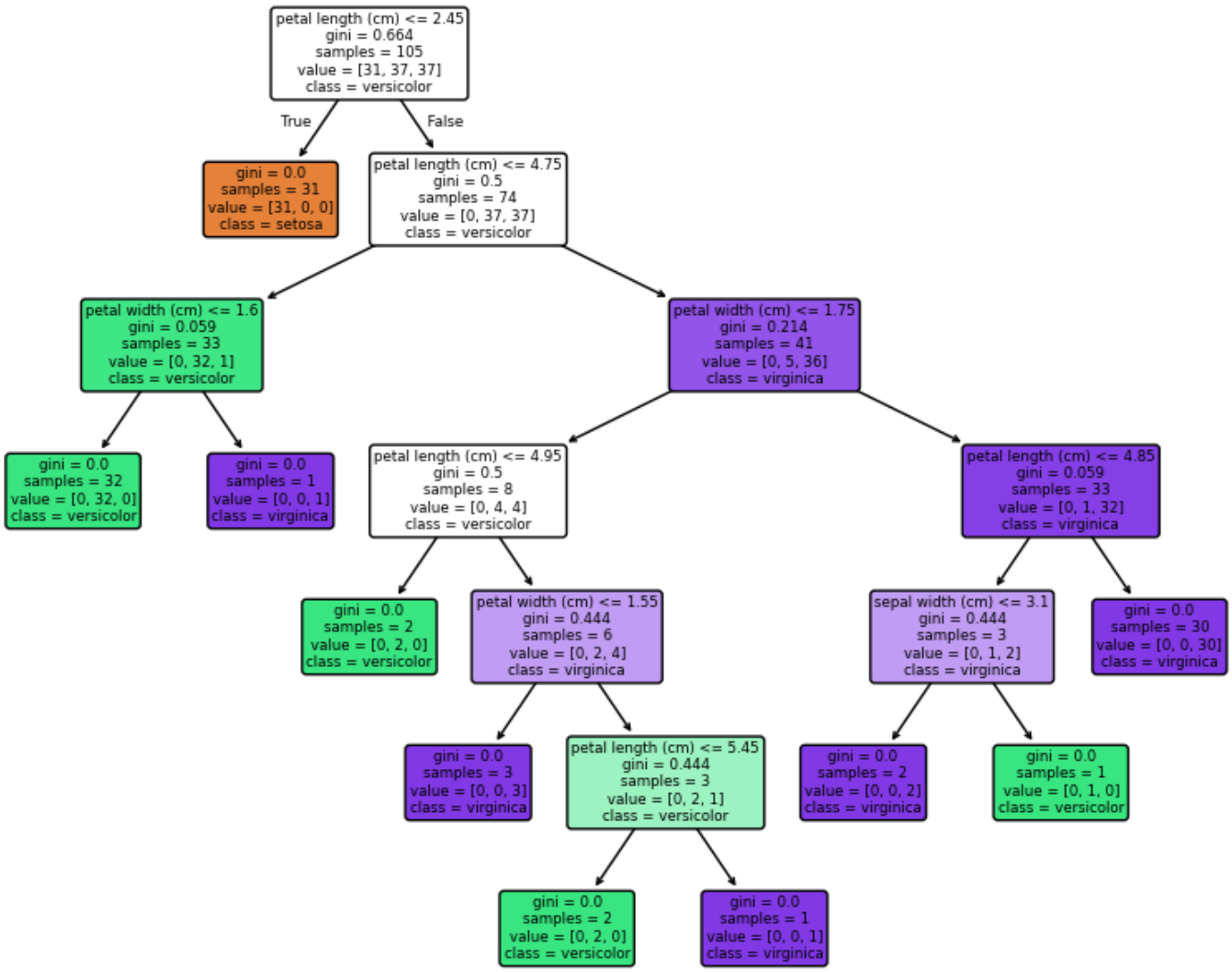
```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy du Decision Tree: {accuracy * 100:.2f}%")
```

```
Accuracy du Decision Tree: 100.00%
```

Visualisation de l'arbre de décision

In [7]:

```
plt.figure(figsize=(10, 8))
tree.plot_tree(clf, filled=True, feature_names=iris.feature_names, class_names=iris.targe
t_names, rounded=True)
plt.show()
```



Classification d'un nouvel échantillon

In [8]:

```
new_sample = [[5.5, 2.4, 3.8, 1.1]]
predicted_class = clf.predict(new_sample)
print(f"Classe prédite pour l'échantillon: {iris.target_names[predicted_class][0]}")
```

```
Classe prédite pour l'échantillon: versicolor
```

Résultats

Précision du modèle

Le modèle a atteint une précision de 100% sur l'ensemble de test. Cela indique que l'arbre de décision est capable de classer parfaitement les échantillons de test.

Visualisation de l'arbre de décision

L'arbre de décision a été visualisé avec succès, montrant les règles de décision utilisées pour classer les échantillons.

Analyse des résultats

Performance

La précision de 100% suggère que l'arbre de décision est très performant sur cet ensemble de données. Cependant, cela pourrait également indiquer un surajustement (overfitting), car l'arbre de décision est capable de mémoriser les données d'entraînement.

Impact des paramètres

Le paramètre random_state=42 garantit la reproductibilité des résultats. Si on modifie ce paramètre, les résultats peuvent varier légèrement. De plus, l'ajustement de la profondeur de l'arbre (via max_depth) pourrait aider à éviter le surajustement.

In [ ]:

```
```