Decision Trees, Random Forests, and Support Vector Machines (SVMs) in Python

In [5]:
```
import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

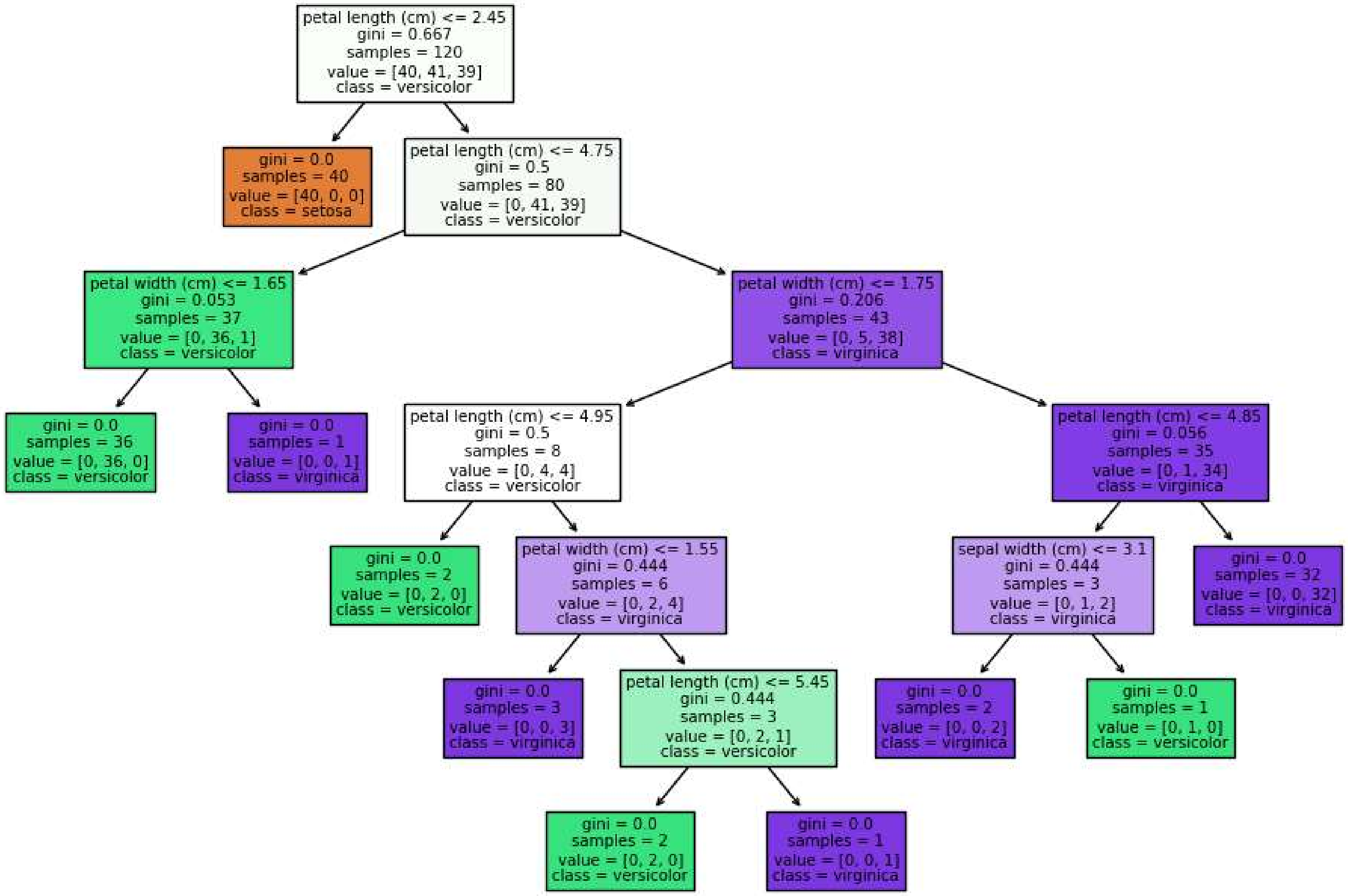
Decision Trees

1. Implement a Decision Tree Classifier
2. Import the required libraries.
3. Load the Iris dataset from sklearn.datasets.
4. Split the data into training and testing sets.
5. Train a DecisionTreeClassifier model.
6. Evaluate the model using accuracy score and visualize the tree.

In [7]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Step 1: Load the dataset
data = load_iris()
X, y = data.data, data.target

# Step 2: Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Train the Decision Tree Classifier
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Step 4: Make predictions
```

```python
predictions = model.predict(X_test)

# Step 5: Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Step 6: Visualize the tree
plt.figure(figsize=(12, 8))
plot_tree(model, feature_names=data.feature_names, class_names=data.target_names, filled=True)
plt.show()
```

```
Accuracy: 100.00%
```



Random Forests

1. Implement a Random Forest Classifier
2. Use the same Iris dataset.
3. Train a RandomForestClassifier model with 100 trees.

4. Evaluate the model using accuracy score.
5. Identify the most important features.

In [9]:
```
from sklearn.ensemble import RandomForestClassifier

# Step 1: Train the Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Step 2: Make predictions
rf_predictions = rf_model.predict(X_test)

# Step 3: Evaluate the model
rf_accuracy = accuracy_score(y_test, rf_predictions)
print(f"Random Forest Accuracy: {rf_accuracy * 100:.2f}%")

# Step 4: Feature importance
importances = rf_model.feature_importances_
for name, importance in zip(data.feature_names, importances):
    print(f"{name}: {importance:.2f}")
```

```
Random Forest Accuracy: 100.00%
sepal length (cm): 0.11
sepal width (cm): 0.03
petal length (cm): 0.44
petal width (cm): 0.42
```

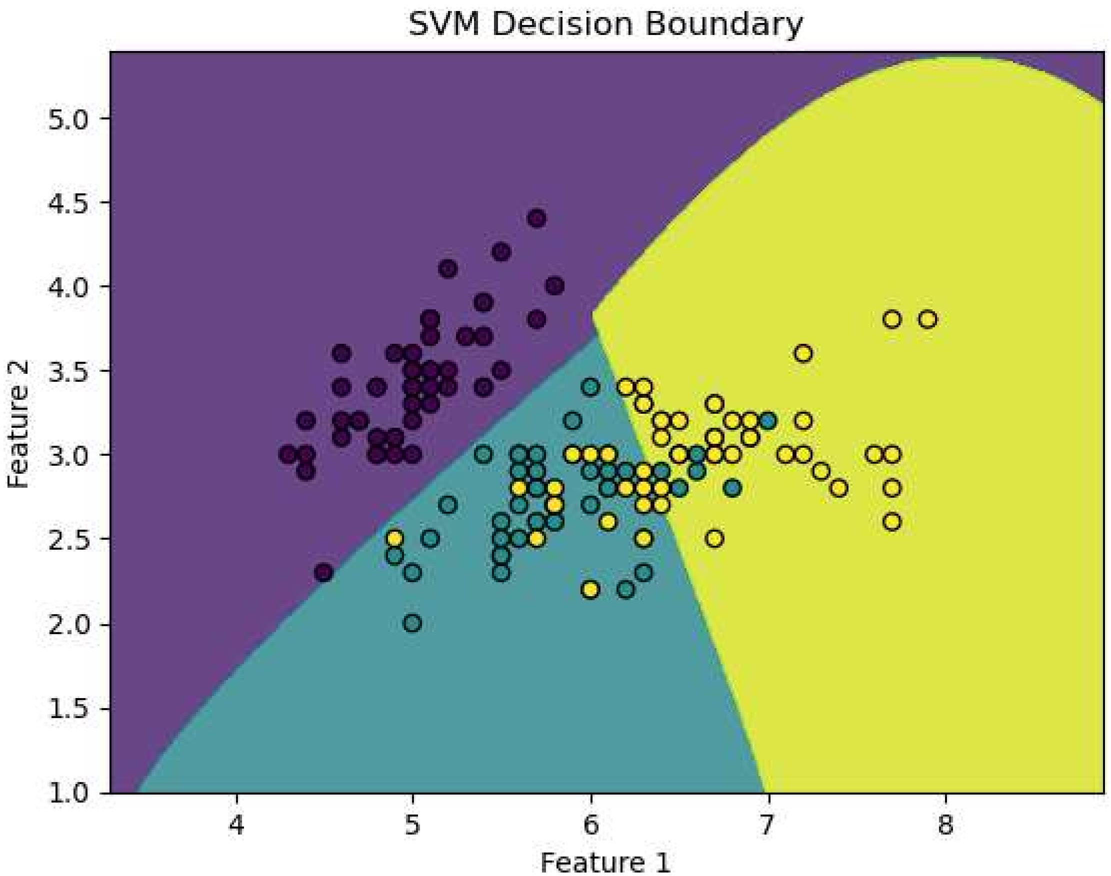
Support Vector Machines (SVMs)

1. Implement a Support Vector Machine Classifier
2. Use the Iris dataset.
3. Train an SVC model with an RBF kernel.
4. Evaluate the model using accuracy score.
5. Visualize the decision boundaries (for two selected features).

In [11]:
```
from sklearn.svm import SVC
import numpy as np

# Step 1: Train the SVM model
```

```python
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
svm_model.fit(X_train, y_train)

# Step 2: Make predictions
svm_predictions = svm_model.predict(X_test)

# Step 3: Evaluate the model
svm_accuracy = accuracy_score(y_test, svm_predictions)
print(f"SVM Accuracy: {svm_accuracy * 100:.2f}%")

# Step 4: Visualize decision boundaries (use only 2 features)
X_subset = X[:, :2]  # Select first two features
X_train_subset, X_test_subset, y_train_subset, y_test_subset = train_test_split(X_subset, y, test_size=0.2, random_st

svm_model_2d = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
svm_model_2d.fit(X_train_subset, y_train_subset)

# Plot decision boundary
def plot_decision_boundary(X, y, model):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max, 0.01))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('SVM Decision Boundary')
    plt.show()

plot_decision_boundary(X_subset, y, svm_model_2d)
```

```
SVM Accuracy: 100.00%
```


SVM Decision Boundary
Feature 2
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8
Feature 1