# Decision Tree Classifier

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.preprocessing import LabelEncoder

# Load the Iris dataset
data = pd.read_csv('/content/iris.csv', skiprows=1)  # Skip the header row

# Manually specify feature names
feature_names = ["sepal_length", "sepal_width", "petal_length", "petal_width", "class"]

# Separate features and target variable
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

# Encode the categorical labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the decision tree classifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)

# Plot the decision tree
plot_tree(classifier, feature_names=feature_names, class_names=label_encoder.classes_, filled=True)

# Perform classification on a new sample
new_sample = [[5.1, 3.5, 1.4, 0.2, 0]]  # Example new sample with a class label

# Print the decision tree rules
def print_decision_rules(tree, feature_names, node=0):
    if tree.feature[node] != -2:
        feature_index = tree.feature[node]
        threshold = tree.threshold[node]
        class_index = tree.value[node].argmax()
        class_label = label_encoder.inverse_transform([class_index])[0]
        rule = f"If {feature_names[feature_index]} <= {threshold} then {class_label}"
        print(rule)

        print_decision_rules(tree, feature_names, node=tree.children_left[node])
        print_decision_rules(tree, feature_names, node=tree.children_right[node])
print_decision_rules(classifier.tree_, feature_names[:-1])  # Exclude the "class" feature
```