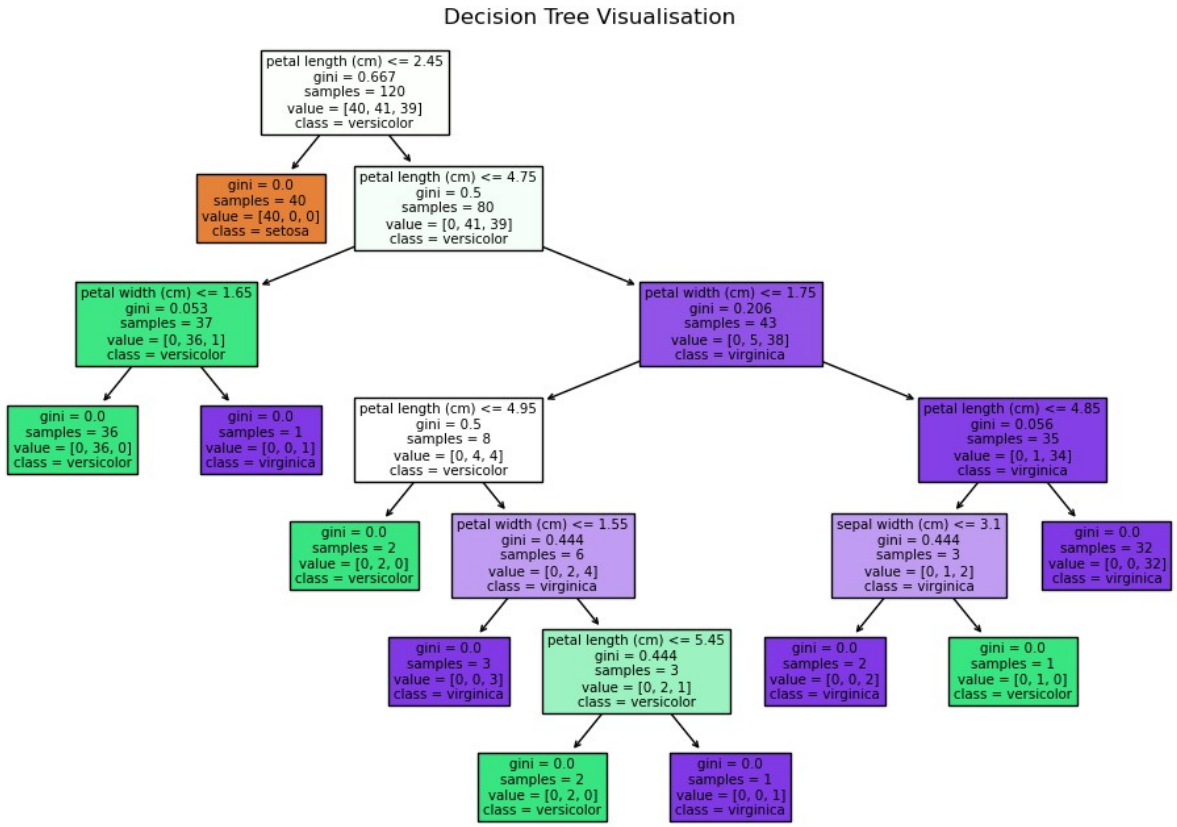
Exp7 Decision Tree

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

iris = load_iris()
X = iris.data  # Features
y = iris.target  # Target labels

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

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

```
DecisionTreeClassifier(random_state=42)
```

```
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=iris.feature_names,
class_names=iris.target_names)
plt.title("Decision Tree Visualisation")
plt.show()
```




```
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy*100:.4f}")
```

```
Accuracy: 100.0000
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred,
average='weighted'))
print("Recall:", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0

Classification Report:
              precision    recall  f1-score   support
```

```
           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Decision Tree Classification on the Iris Dataset

Name: Viraj Mulik
Roll No: A27
Batch: A1

---

1. Introduction

The Iris dataset is a well-known dataset in machine learning, consisting of 150 samples with four features describing the sepal and petal dimensions of three species of Iris flowers. For this experiment, we employed the Decision Tree Classifier, a versatile supervised learning algorithm, to classify the species of the flowers.

---

2. Dataset Preparation

The dataset was loaded using scikit-learn's load_iris function. It was divided into features (X) and target labels (y) and further split into training and testing sets using an 80-20 split.

---

3. Model Training

We used the **DecisionTreeClassifier** from scikit-learn with the default parameters. The training data was used to fit the model, which was later evaluated on the testing set.

The visualization of the decision tree provided insights into the decision rules derived from the training data. The tree's nodes represent splits, while the leaves indicate predictions.

---

4. Model Evaluation

The trained model was evaluated on the test set using the following metrics:

Metric	Score
Accuracy	100%
Precision	100%
Recall	100%
F1-Score	100%

Confusion Matrix

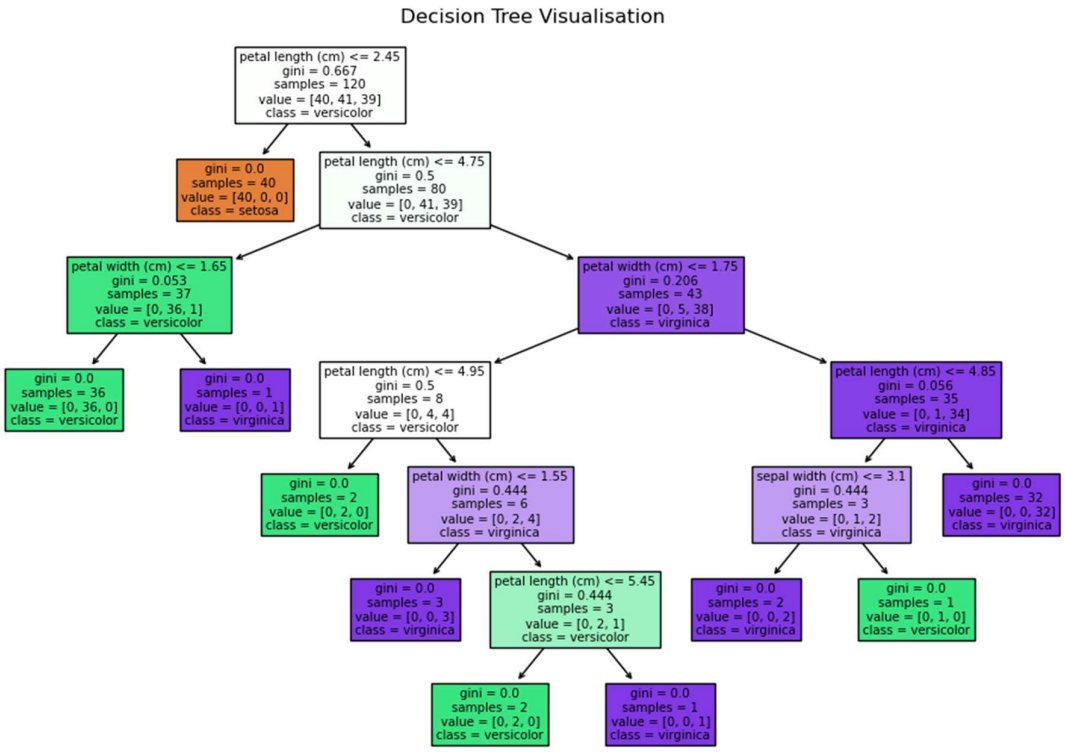
```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

These results indicate perfect classification on the test set, but it raises concerns about potential overfitting due to the small size of the dataset.

---

5. Visualization

Below is a graphical representation of the decision tree structure:



p

---

6. Analysis

The perfect scores across all metrics suggest overfitting, a common issue with decision trees trained to full depth on small datasets. Regularization techniques, such as limiting max_depth, could help mitigate this.

---

7. Conclusion

The Decision Tree Classifier effectively classified the Iris dataset, achieving 100% accuracy on the test set. However, for larger or more complex datasets, careful tuning of hyperparameters like max_depth is essential to avoid overfitting and ensure better generalization.