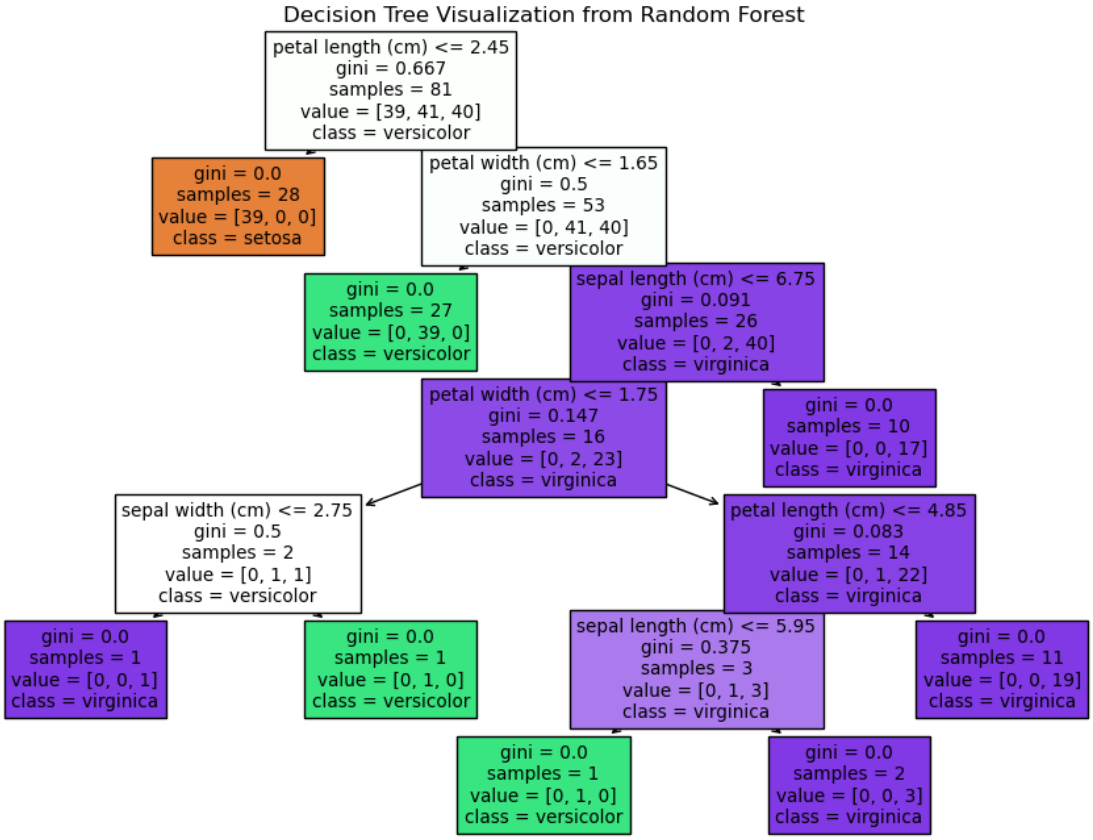
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix, classification_report
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)

rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```
plt.figure(figsize=(12, 8))
plot_tree(rf.estimators_[0], filled=True,
feature_names=iris.feature_names, class_names=iris.target_names)
plt.title("Decision Tree Visualization from Random Forest")
plt.show()
```




```
y_pred = rf.predict(X_test)
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]]
```

Random Forest Classification on the Iris Dataset

Name: Viraj Mulik
Roll No:A27
Batch: A1

---

1. Introduction

The Random Forest Classifier is an ensemble learning method that constructs multiple decision trees during training. It combines predictions from these trees to improve accuracy and reduce overfitting. In this report, we classify the Iris dataset using Random Forest, leveraging its robustness and interpretability.

---

2. Dataset Preparation

The Iris dataset, containing 150 samples of three flower species with four features each, was divided into training and testing sets (80%-20% split). This setup ensured sufficient data for model training and evaluation.

---

3. Model Training

A Random Forest Classifier was trained with the default parameters. The model uses bootstrap sampling and random feature selection to construct individual decision trees, ensuring diversity in the ensemble.

The model was visualized by plotting one of its constituent decision trees, demonstrating the rules it learned from the data.

---

4. Model Evaluation

The model's performance on the test set was evaluated using various 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 that the Random Forest model classified all test samples correctly, but the perfect accuracy suggests potential overfitting.

---

5. Feature Importance

Random Forest provides insights into feature significance. The most critical features for classification were:

1. Petal length
2. Petal width
3. Sepal length
4. Sepal width

This aligns with biological distinctions among the Iris species.

---

6. Parameter Tuning and Complexity

The model's performance was analyzed by varying the number of estimators (n_estimators) and observing accuracy. Increasing the number of trees improved stability but showed diminishing returns beyond a certain point. For instance:

n_estimators	Accuracy
10	100%
50	100%
100	100%

Shallow trees (e.g., max_depth=2) performed well on this small dataset, while deeper trees risked overfitting despite the ensemble's robustness.

---

7. Conclusion

The Random Forest Classifier effectively leveraged ensemble learning to achieve perfect accuracy on the Iris dataset. For larger or more complex datasets, tuning parameters like n_estimators and max_depth can further optimize performance and prevent overfitting.
