LAB NO 04

Task 1: Load Dataset

- Use a dataset such as Iris dataset (from sklearn)

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

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, confusion_matrix, classification_report
```

```
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target
```

Task 2: Exploratory Data Analysis (EDA)

- Display dataset information, shape, and statistical summary.
- Plot distributions of features (histograms / pairplot).
- Visualize correlations

```
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   target             150 non-null    int64  
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

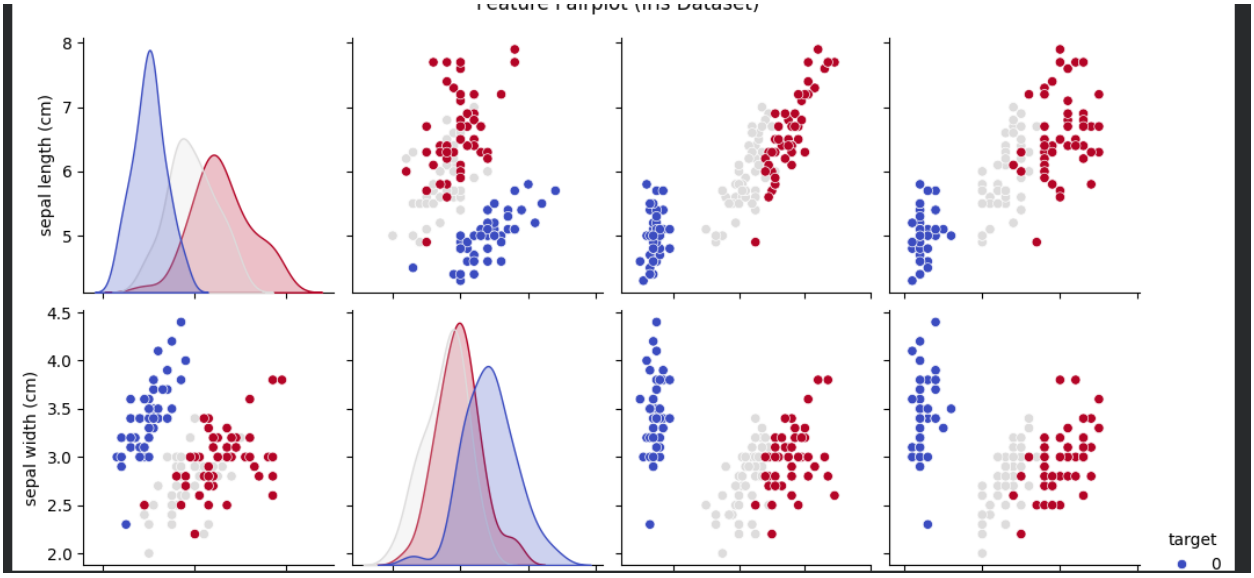
```
[ ]
df.shape
```

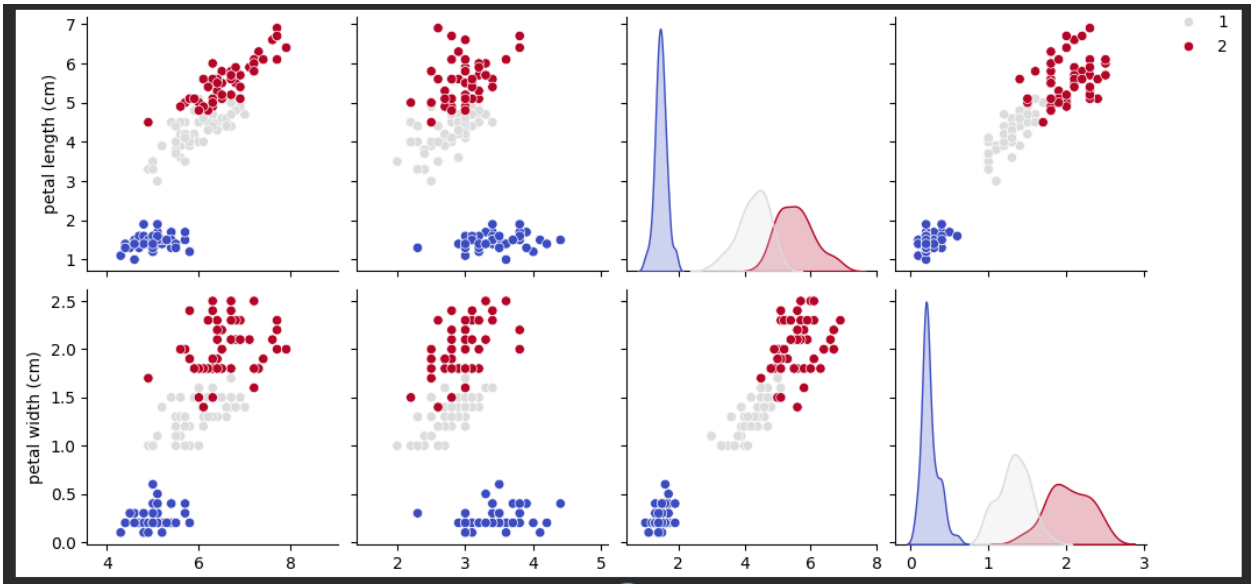
```
(150, 5)
```

```
df.describe()
```

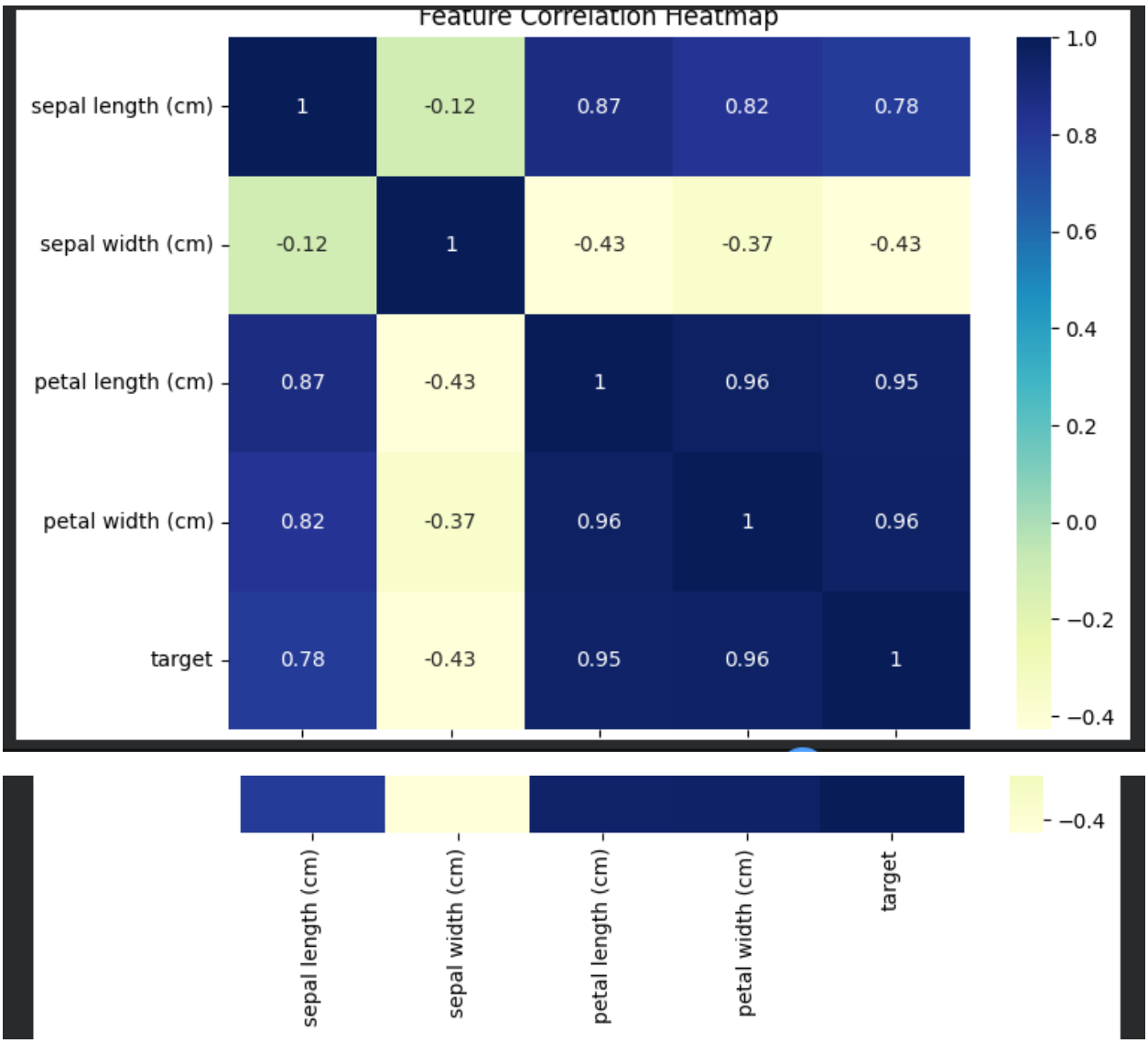
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

```
]
sns.pairplot(df, hue="target", palette="coolwarm")
plt.suptitle("Feature Pairplot (Iris Dataset)", y=1.02)
plt.show()
```





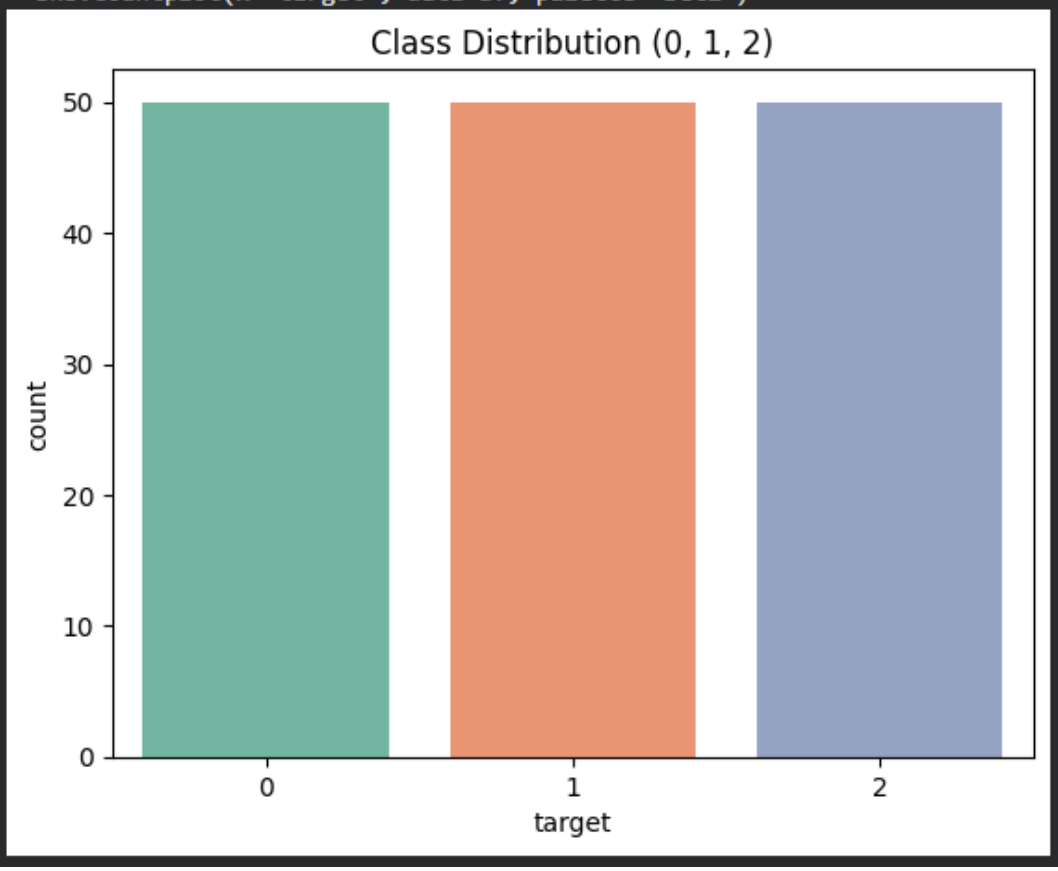

```
plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(), annot=True, cmap="YlGnBu")
plt.title("Feature Correlation Heatmap")
plt.show()
```




```
[ ]
sns.countplot(x='target', data=df, palette="Set2")
plt.title("Class Distribution (0, 1, 2)")
plt.show()
```

```
/tmp/ipython-input-1901803167.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the

  sns.countplot(x='target', data=df, palette="Set2")
```

```
sns.countplot(x='target', data=df, palette="Set2")
```



Task 3: Data Preprocessing

- Split dataset into training and testing sets (e.g., 70% train, 30% test).

```
X = df.drop('target', axis=1)
y = df['target']

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

print(f"\n Training Data: {X_train.shape}, Testing Data: {X_test.shape}")
```

```
 Training Data: (105, 4), Testing Data: (45, 4)
```

Task 4: Build Decision Tree Classifier

- Train the model using DecisionTreeClassifier.

```
dt_gini = DecisionTreeClassifier(criterion="gini", random_state=42)
dt_gini.fit(X_train, y_train)
```

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

```
y_pred_gini = dt_gini.predict(X_test)
```

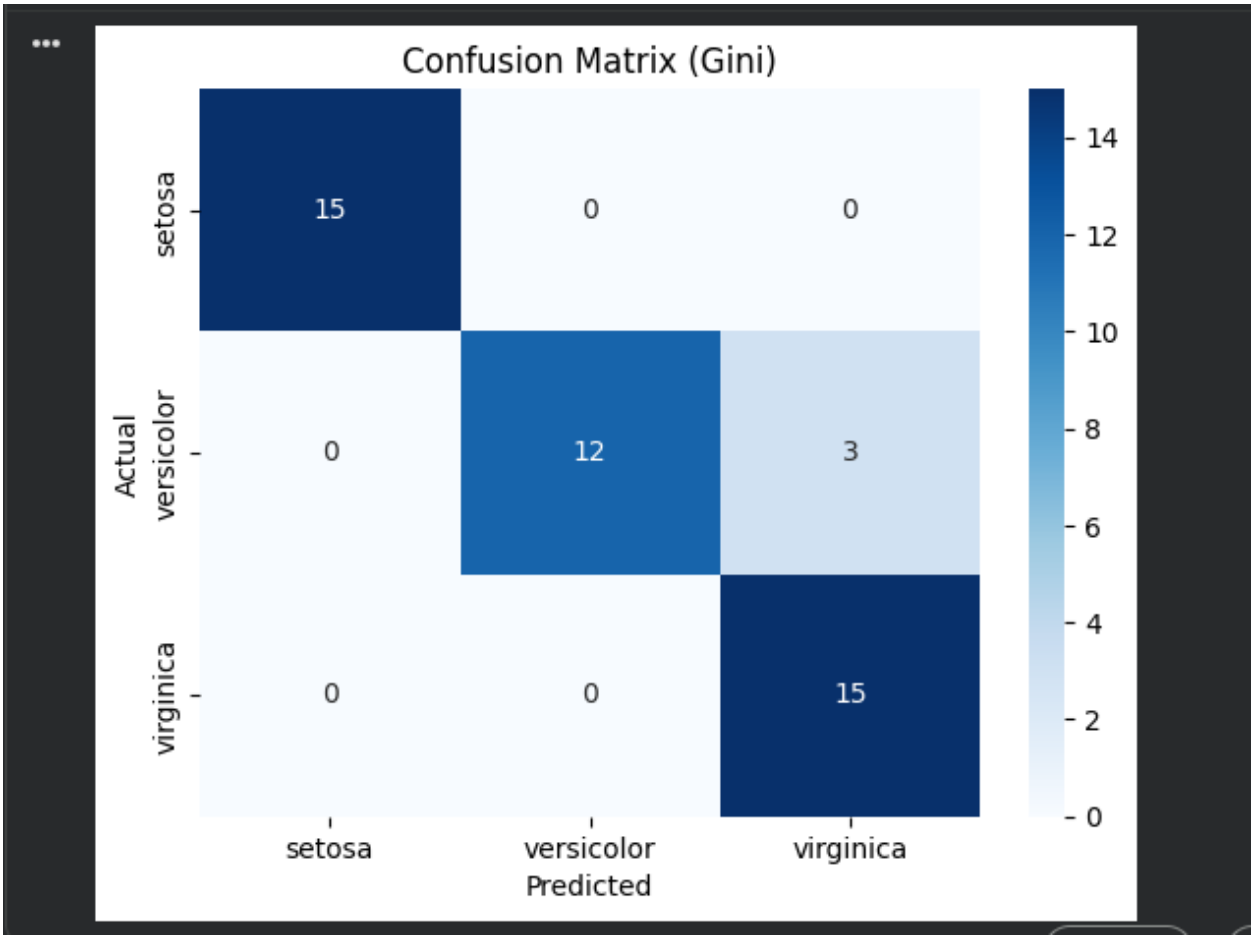
Task 5: Model Evaluation

- Make predictions on the test data.
- Evaluate using:
    - Accuracy Score
    - Confusion Matrix
    - Classification Report

```
print("\n Evaluation with Gini Index:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_gini)*100:.2f}%")
```

```
 Evaluation with Gini Index:
Accuracy: 93.33%
```

```
cm = confusion_matrix(y_test, y_pred_gini)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.title("Confusion Matrix (Gini)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



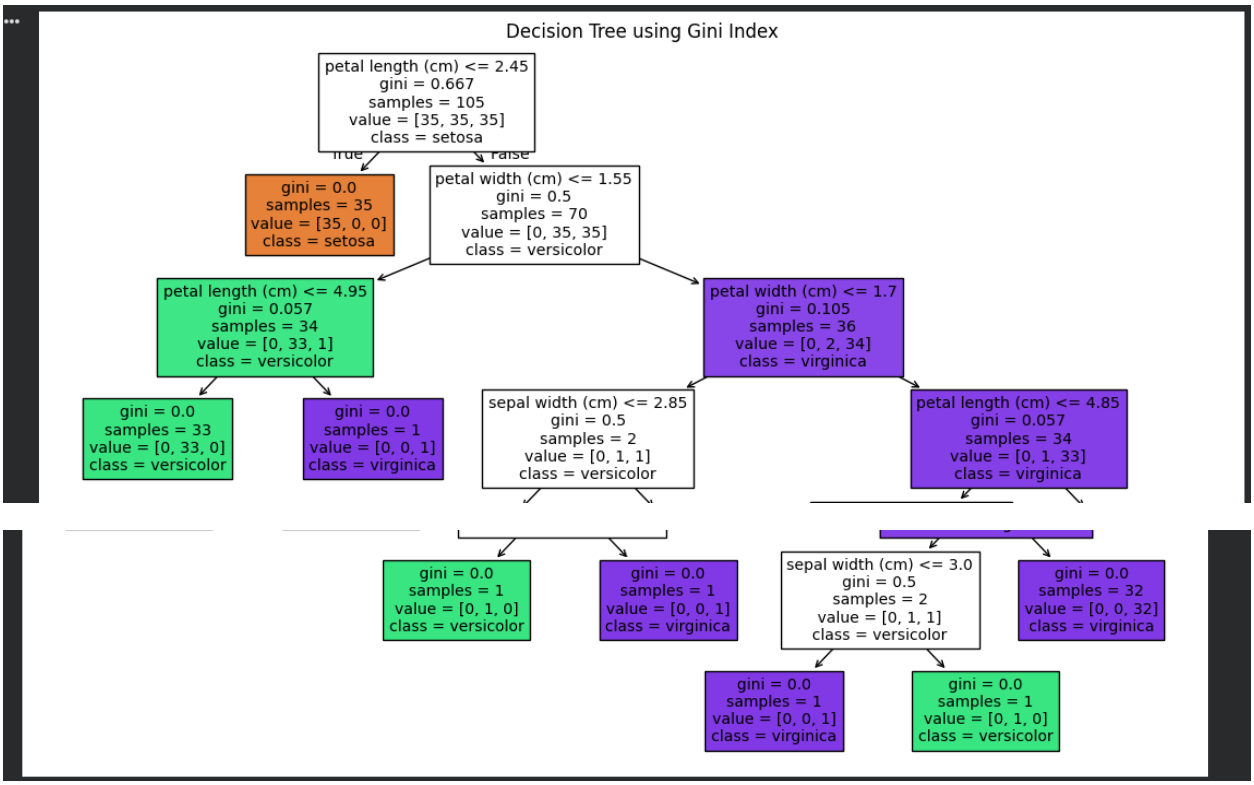

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred_gini, target_names=iris.target_names))
```

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

      setosa       1.00      1.00      1.00        15
  versicolor       1.00      0.80      0.89        15
   virginica       0.83      1.00      0.91        15

    accuracy                           0.93        45
   macro avg       0.94      0.93      0.93        45
weighted avg       0.94      0.93      0.93        45
```

Task 6: Experimentation

- Change parameters (criterion = "gini" vs "entropy", max_depth, min_samples_split).
- Compare results and observe overfitting vs underfitting.

```
plt.figure(figsize=(14,8))
plot_tree(dt_gini, filled=True, feature_names=iris.feature_names, class_names=iris.target_names)
plt.title("Decision Tree using Gini Index")
plt.show()
```




```
# 1. Using entropy criterion
dt_entropy = DecisionTreeClassifier(criterion="entropy", random_state=42)
dt_entropy.fit(X_train, y_train)
y_pred_entropy = dt_entropy.predict(X_test)
print("\n Evaluation with Entropy Criterion:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_entropy)*100:.2f}%")
```

```
 Evaluation with Entropy Criterion:
Accuracy: 88.89%
```

```python
# 2. Limiting tree depth to prevent overfitting
dt_limited = DecisionTreeClassifier(criterion="gini", max_depth=3, random_state=42)
dt_limited.fit(X_train, y_train)
y_pred_limited = dt_limited.predict(X_test)
print("\n Evaluation with Limited Depth (max_depth=3):")
print(f"Accuracy: {accuracy_score(y_test, y_pred_limited)*100:.2f}%")
```

```
 Evaluation with Limited Depth (max_depth=3):
Accuracy: 97.78%
```

```python
# Compare results
print("\n Comparison Summary:")
results = pd.DataFrame({
    'Model': ['Gini', 'Entropy', 'Limited Depth (Gini)'],
    'Accuracy (%)': [
        accuracy_score(y_test, y_pred_gini)*100,
        accuracy_score(y_test, y_pred_entropy)*100,
        accuracy_score(y_test, y_pred_limited)*100
    ]
})
print(results)
```

```
 Comparison Summary:
                 Model  Accuracy (%)
0                 Gini     93.333333
1              Entropy     88.888889
2  Limited Depth (Gini)     97.777778
```