Roll Number:- 22102B2001

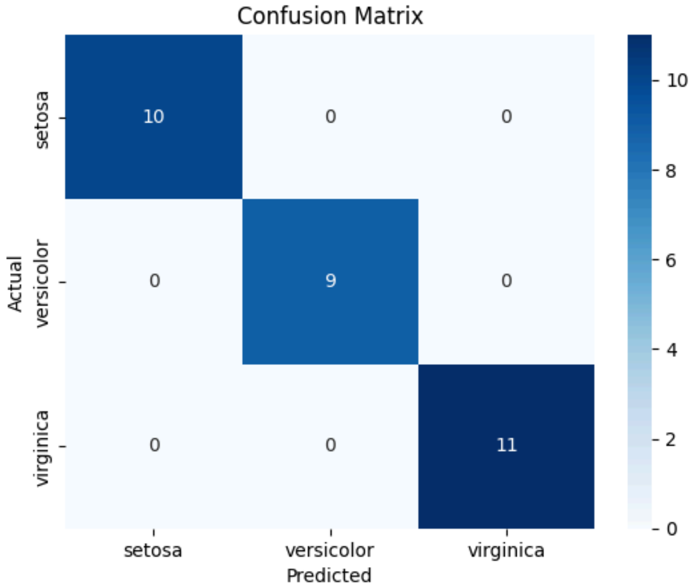
Name:- Chinmay Mhatre

Github Link:- https://github.com/chinmay0910/ML-Lab---VIT

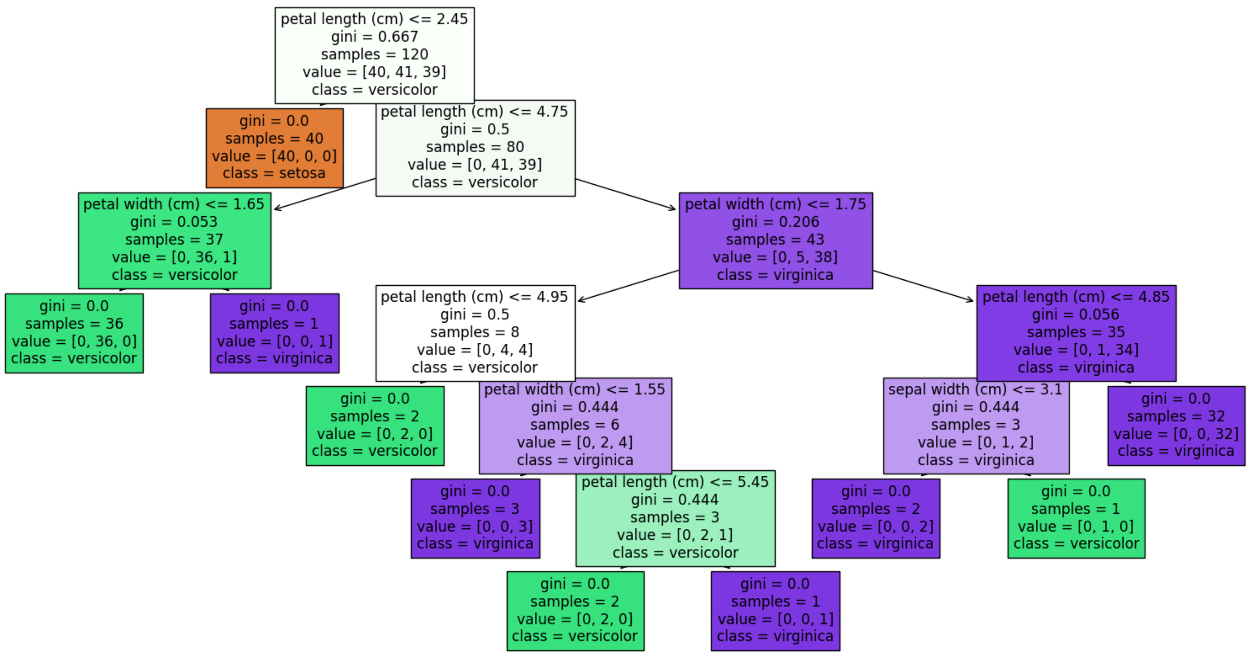
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, roc_auc_score, roc_curve, cohen_kappa_score, precisi

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the data into training and testing 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 using the Gini Index
clf = DecisionTreeClassifier(criterion='gini', random_state=42)
clf.fit(X_train, y_train)

# Predict the target variable on the testing set
y_pred = clf.predict(X_test)

# Evaluate the classifier's performance
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_matrix)

# Plot the Confusion Matrix
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

print("Classification Report:\n", classification_report(y_test, y_pred))
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Calculate and interpret the Kappa Statistics
kappa = cohen_kappa_score(y_test, y_pred)
print(f"Kappa Statistics: {kappa}")

# Calculate Sensitivity, Specificity, Precision, Recall, and F-measure for each class
precision, recall, f1, _ = precision_recall_fscore_support(y_test, y_pred, average=None)
sensitivity = recall  # Sensitivity is the same as recall for each class
specificity = [conf_matrix[i, i] / sum(conf_matrix[:, i]) for i in range(len(conf_matrix))]

# Macro average
macro_precision = np.mean(precision)
macro_recall = np.mean(recall)
macro_f1 = np.mean(f1)
macro_specificity = np.mean(specificity)

print(f"Sensitivity (Recall): {sensitivity}")
print(f"Specificity: {specificity}")
print(f"Precision: {precision}")
print(f"F-measure: {f1}")

print(f"Macro Precision: {macro_precision}")
print(f"Macro Recall: {macro_recall}")
print(f"Macro F1: {macro_f1}")
print(f"Macro Specificity: {macro_specificity}")

# Plot the Decision Tree
plt.figure(figsize=(20,10))
plot_tree(clf, filled=True, feature_names=iris.feature_names, class_names=iris.target_names)
plt.show()

# ROC Curve and AUC
y_prob = clf.predict_proba(X_test)
roc_auc = roc_auc_score(y_test, y_prob, multi_class='ovo')
```

```
roc_auc = roc_auc_score(y_test, y_prob, multi_class="ovo")
print(f"AUC: {roc_auc}")
# For multiclass ROC curve, we need to plot ROC curve for each class separately
fpr = {}
tpr = {}
for i in range(3):
    fpr[i], tpr[i], _ = roc_curve(y_test, y_prob[:, i], pos_label=i)
    plt.plot(fpr[i], tpr[i], label=f"Class {i} ROC Curve")

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```

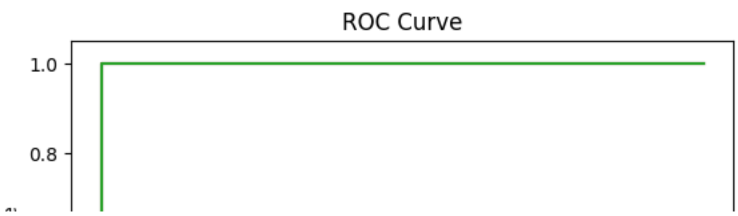
```
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

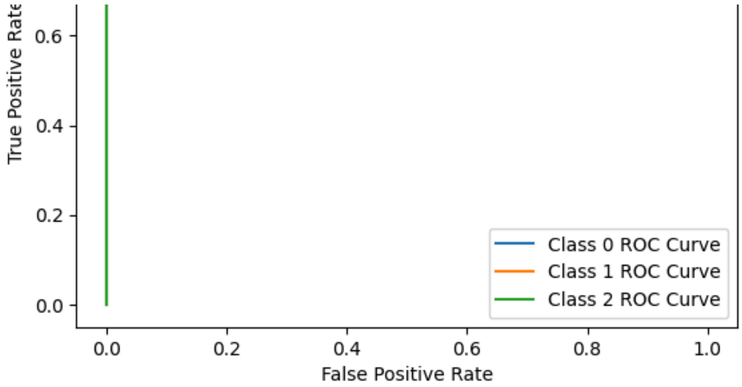



```
Classification Report:
              precision    recall  f1-score   support

           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

Accuracy: 1.0
Kappa Statistics: 1.0
Sensitivity (Recall): [1. 1. 1.]
Specificity: [1.0, 1.0, 1.0]
Precision: [1. 1. 1.]
F-measure: [1. 1. 1.]
Macro Precision: 1.0
Macro Recall: 1.0
Macro F1: 1.0
Macro Specificity: 1.0
```




```
AUC: 1.0
```






```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load the Boston Housing dataset
url = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"
data = pd.read_csv(url)

# Split the data into training and testing sets
X = data.drop('medv', axis=1)
y = data['medv']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the Decision Tree regressor using mean squared error
reg = DecisionTreeRegressor(criterion='squared_error', random_state=42)
reg.fit(X_train, y_train)

# Predict the target variable on the testing set
y_pred = reg.predict(X_test)

# Evaluate the regressor's performance
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"MSE: {mse}")
print(f"MAE: {mae}")
print(f"R-squared: {r2}")

# Plot actual vs. predicted values with two different colors
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted Values')
plt.scatter(y_test, y_test, color='red', label='Actual Values')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.legend()
plt.show()
```