EXP NO: 3	**DECISION TREE CLASSIFIER**
DATE: 14/08/2025	

AIM:

To implement a Decision Tree classifier and evaluate its performance using **accuracy score** and **confusion matrix** on a real-world dataset.

ALGORITHM:

STEP 1: Import necessary libraries

STEP 2: Load a **classification dataset** (e.g., Iris or Titanic)

STEP 3: Split the dataset into training and test sets

STEP 4: Preprocess data if needed

STEP 5: Train a **DecisionTreeClassifier** from **sklearn.tree**

STEP 6: Predict on test data

STEP 7: Evaluate using:

- **Confusion Matrix**
- **Accuracy Score**

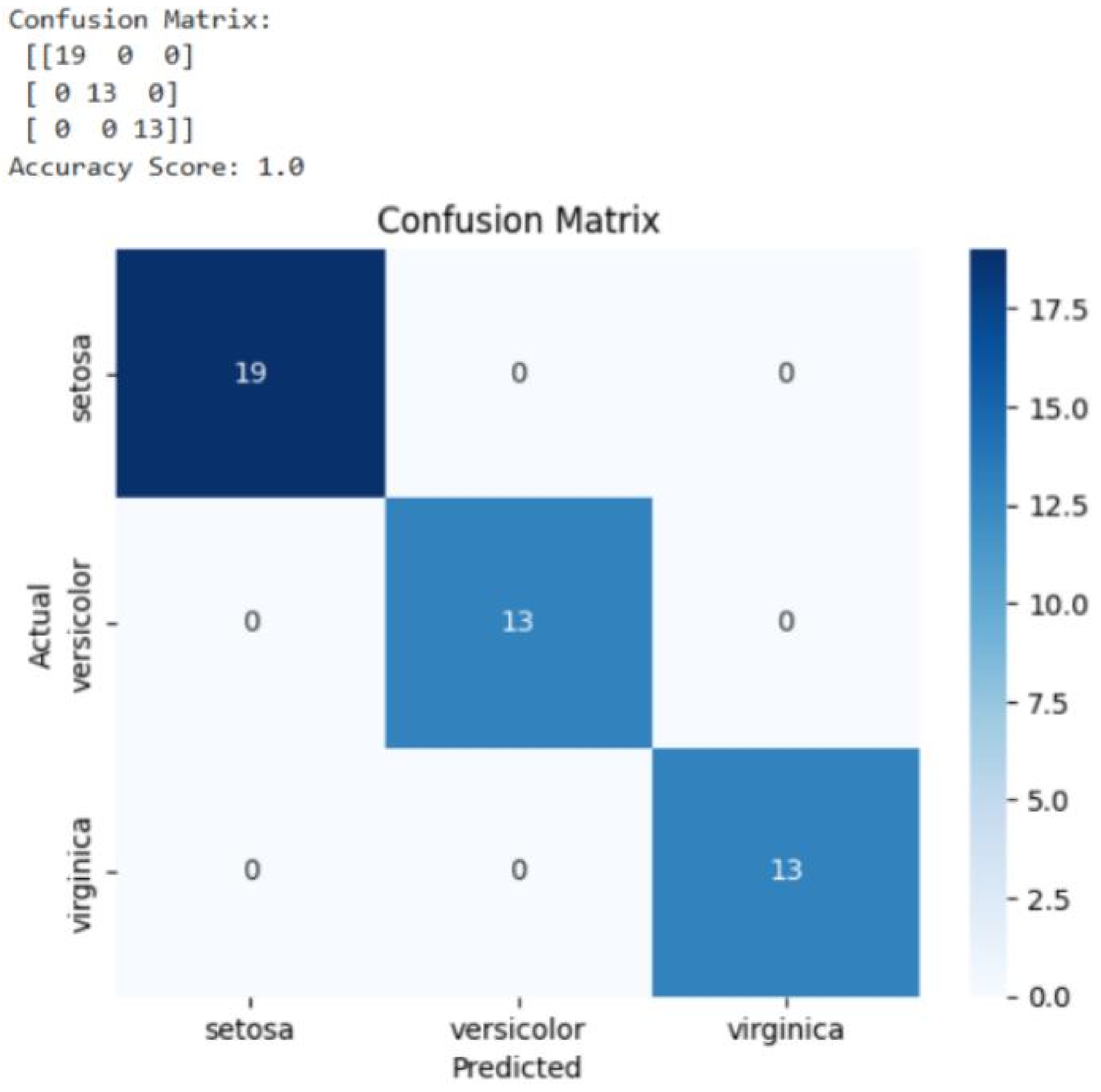
STEP 8: Visualize the **Decision Tree** (optional)

CODE:

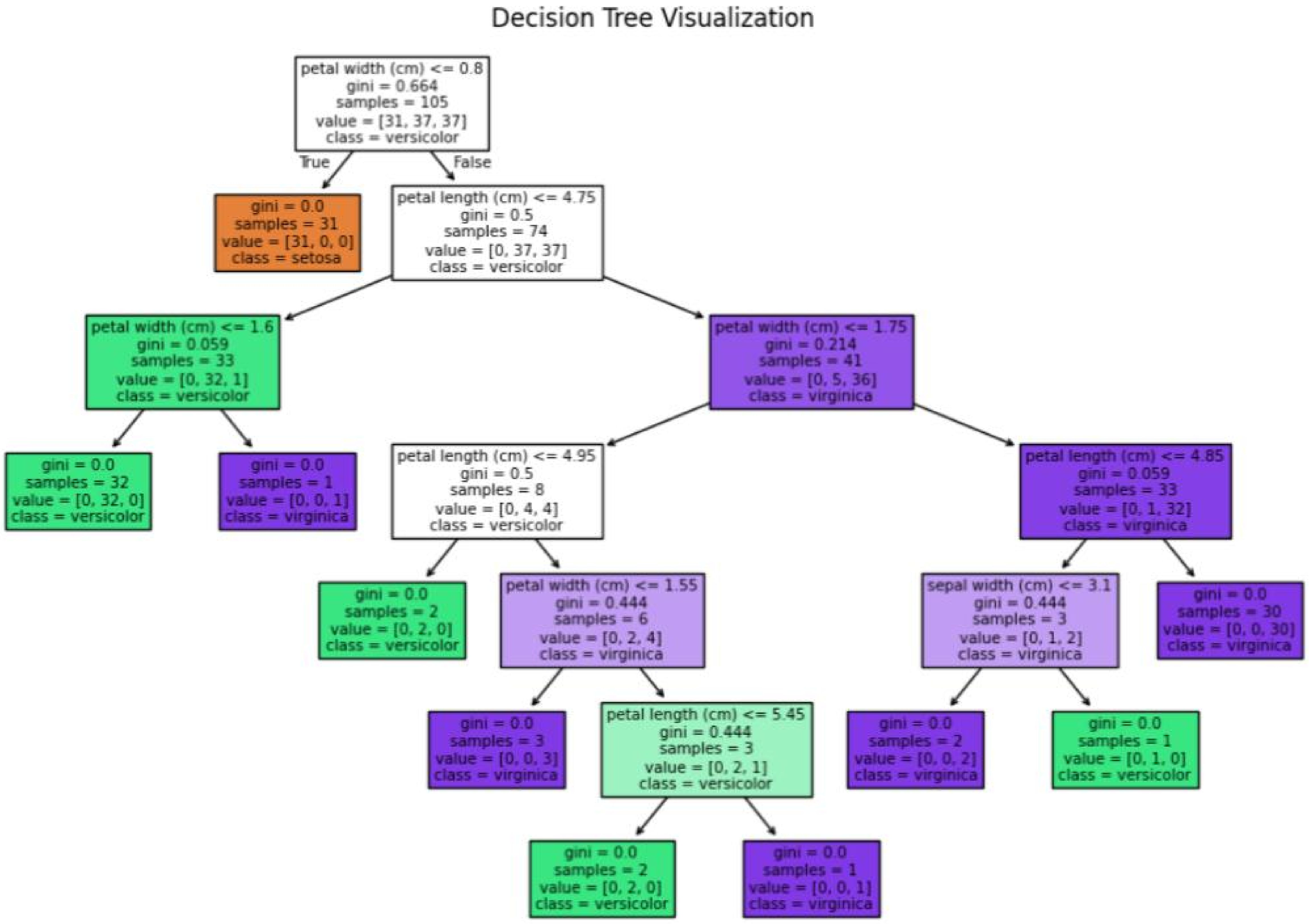
```
# Step 1: Import Libraries
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
```

```
import matplotlib.pyplot as plt
import seaborn as sns


# Step 2: Load Dataset
iris = load_iris()
X = iris.data
y = iris.target

# Step 3: Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Step 4: Train the Decision Tree Classifier
dt_model = DecisionTreeClassifier(criterion='gini', random_state=0)
dt_model.fit(X_train, y_train)

# Step 5: Predict
y_pred = dt_model.predict(X_test)

# Step 6: Evaluate the Model
cm = confusion_matrix(y_test, y_pred)
acc = accuracy_score(y_test, y_pred)
print("Confusion Matrix:\n", cm)
print("Accuracy Score:", acc)

# Step 7: Visualize Confusion Matrix
sns.heatmap(cm, annot=True, cmap="Blues", xticklabels=iris.target_names,
yticklabels=iris.target_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Step 8: Visualize the Decision Tree
plt.figure(figsize=(12,8))
plot_tree(dt_model, filled=True, feature_names=iris.feature_names,
class_names=iris.target_names)
plt.title("Decision Tree Visualization")
plt.show()
```

OUTPUT:



```
Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
Accuracy Score: 1.0
```



RESULT:

Thus, the execution successfully implemented **SVM** and **Random Forest** classification models, applied them to the given dataset, and evaluated their performance using **accuracy** and **confusion matrix** to measure classification effectiveness.