Program 7: Solving Classification problem using Decision tree

Problem Statement:

The goal of this program is to build and evaluate a Decision Tree classifier using the Breast Cancer dataset to predict whether a tumor is benign or malignant. The program will load the dataset, explore its features, train a Decision Tree model, evaluate its performance, and visualize the resulting decision tree.

```
from sklearn.datasets import load_breast_cancer  # Import function to load the Breast Cancer dataset
from sklearn.model_selection import train_test_split  # Import function to split data into training and testing sets
from sklearn.tree import DecisionTreeClassifier  # Import Decision Tree classifier
from sklearn.metrics import accuracy_score, classification_report  # Import functions to evaluate model performance
from sklearn import tree  # Import module for visualizing decision trees
import matplotlib.pyplot as plt  # Import plotting library

# Load Breast Cancer dataset
data = load_breast_cancer()  # Load dataset and store in 'data' variable
X = data.data  # Extract feature data (e.g., measurements) into 'X'
y = data.target  # Extract target labels (e.g., benign or malignant) into 'y'

print("Feature names:", data.feature_names)  # Print names of features
print("Class names:", data.target_names)  # Print names of target classes
print("First two rows of the dataset:")  # Print message indicating the start of dataset preview
print(X[:2])  # Print the first two rows of feature data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)  # Split data into 70% training and 30% testing; set seed for reproducibility

clf = DecisionTreeClassifier()  # Create an instance of the Decision Tree classifier
clf.fit(X_train, y_train)  # Train the classifier on the training data
```

```
y_pred = clf.predict(X_test)  # Predict labels for the test data
accuracy = accuracy_score(y_test, y_pred)  # Compute the accuracy of the predictions
print("Accuracy:", accuracy)  # Print the accuracy score
print("Classification Report:")  # Print message indicating the start of classification report
print(classification_report(y_test, y_pred, target_names=data.target_names))  # Print detailed performance metrics


plt.figure(figsize=(20,10))  # Create a new figure with specified size (20x10 inches)
tree.plot_tree(clf, feature_names=data.feature_names, class_names=data.target_names, filled=True)  # Plot the decision tree with feature names and class names
plt.show()  # Display the plot
```