Python code to implement bagging using Random Forest and boosting using AdaBoost and Gradient Boosting.

Explanation

Loading Dataset: load_iris(): Loads the Iris dataset, a popular dataset for classification tasks. X and y: Features and target variables.

Splitting Dataset: train_test_split(): Splits the dataset into training and testing sets with a specified test size (e.g., 20%).

Model Initialization: RandomForestClassifier(): Initializes the Random Forest model with a specified number of estimators. DecisionTreeClassifier(max_depth=1): Initializes a simple decision tree (decision stump) for AdaBoost. AdaBoostClassifier(): Initializes the AdaBoost model with the decision stump as the base estimator and a specified number of estimators. GradientBoostingClassifier(): Initializes the Gradient Boosting model with specified parameters like the number of estimators, learning rate, and max depth.

Training the Model: fit(): Trains the model on the training data.

Making Predictions: predict(): Makes predictions on the testing data.

Evaluating the Model: accuracy_score(): Calculates the accuracy of the model. classification_report(): Provides a detailed classification report with precision, recall, and F1-score. confusion_matrix(): Generates the confusion matrix to visualize the performance.

Bagging with Random Forest

Step-by-Step Implementation Import Libraries: Load necessary libraries. Load Dataset: Use a sample dataset from scikit-learn. Split Dataset: Split the data into training and testing sets. Initialize and Train Model: Set up the Random Forest model and train it. Predict and Evaluate: Make predictions and evaluate the model.

In [4]:
```
# Import necessary libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.datasets import load_iris
import pandas as pd

# Load dataset
data = load_iris()
X = data.data
y = data.target
```

```
# Display dataset information
df = pd.DataFrame(X, columns=data.feature_names)
df['target'] = y
print("First 5 rows of the dataset:")
print(df.head())

# Split the dataset 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)

# Initialize the RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Random Forest Accuracy: {accuracy:.2f}')

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
First 5 rows of the dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0
Random Forest Accuracy: 1.00

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]]
```

Boosting with AdaBoost

Step-by-Step Implementation Import Libraries: Load necessary libraries. Load Dataset: Use a sample dataset from scikit-learn. Split Dataset: Split the data into training and testing sets. Initialize and Train Model: Set up the AdaBoost model with a weak learner (decision stump) and train it. Predict and Evaluate: Make predictions and evaluate the model.

In [5]:
```
# Import necessary libraries
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.datasets import load_iris
```

```
import pandas as pd

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

# Display dataset information
df = pd.DataFrame(X, columns=data.feature_names)
df['target'] = y
print("First 5 rows of the dataset:")
print(df.head())

# Split the dataset 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)

# Initialize the base estimator (decision stump)
base_estimator = DecisionTreeClassifier(max_depth=1)

# Initialize the AdaBoostClassifier
model = AdaBoostClassifier(base_estimator=base_estimator, n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'AdaBoost Accuracy: {accuracy:.2f}')

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
First 5 rows of the dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0
F:\Users\chaitra\anaconda3\Lib\site-packages\sklearn\ensemble\_base.py:156: FutureWarning: `base_estimator` was renamed
to `estimator` in version 1.2 and will be removed in 1.4.
  warnings.warn(
AdaBoost Accuracy: 1.00

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]]
```

Boosting with Gradient Boosting

Step-by-Step Implementation Import Libraries: Load necessary libraries. Load Dataset: Use a sample dataset from scikit-learn. Split Dataset: Split the data into training and testing sets. Initialize and Train Model: Set up the Gradient Boosting model and train it. Predict and Evaluate: Make predictions and evaluate the model.

In [6]:
```
# Import necessary libraries
from sklearn.ensemble import GradientBoostingClassifier
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.datasets import load_iris
import pandas as pd

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

# Display dataset information
df = pd.DataFrame(X, columns=data.feature_names)
df['target'] = y
print("First 5 rows of the dataset:")
print(df.head())

# Split the dataset 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)

# Initialize the GradientBoostingClassifier
model = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Gradient Boosting Accuracy: {accuracy:.2f}')

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
First 5 rows of the dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0
Gradient Boosting Accuracy: 1.00

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]]
```

In [ ]: