In [1]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
```

In [2]:
```
# Load dataset
iris = load_iris()
X = iris.data
y = iris.target
```

In [3]:
```
print(iris)
```

```
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
```

In [4]:
```
# Split data menjadi training dan testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Buat model Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [5]:
```
# Latih model
rf.fit(X_train, y_train)

# Evaluasi model
accuracy = rf.score(X_test, y_test)
print("Akurasi:", accuracy)
```

```
Akurasi: 1.0
```

In [6]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [10, 50, 100, 200],
    'max_depth': [None, 5, 10, 15]
}
```

In [7]:
```
grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Hyperparameter terbaik:", grid_search.best_params_)
print("Akurasi terbaik:", grid_search.best_score_)
```

```
Hyperparameter terbaik: {'max_depth': None, 'n_estimators': 10}
Akurasi terbaik: 0.95
```

In [8]:
```
y_pred=rf.predict(X_test)
print(y_pred)
```

```
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
```

In [9]:

```python
import pandas as pd

# Buat dataframe untuk hasil prediksi
df_pred = pd.DataFrame({
    'X1': X_test[:, 0],
    'X2': X_test[:, 1],
    'X3': X_test[:, 2],
    'X4': X_test[:, 3],
    'y_actual': y_test,
    'y_pred': y_pred
})

# Cetak dataframe hasil prediksi
print(df_pred)
```

	X1	X2	X3	X4	y_actual	y_pred
0	6.1	2.8	4.7	1.2	1	1
1	5.7	3.8	1.7	0.3	0	0
2	7.7	2.6	6.9	2.3	2	2
3	6.0	2.9	4.5	1.5	1	1
4	6.8	2.8	4.8	1.4	1	1
5	5.4	3.4	1.5	0.4	0	0
6	5.6	2.9	3.6	1.3	1	1
7	6.9	3.1	5.1	2.3	2	2
8	6.2	2.2	4.5	1.5	1	1
9	5.8	2.7	3.9	1.2	1	1
10	6.5	3.2	5.1	2.0	2	2
11	4.8	3.0	1.4	0.1	0	0
12	5.5	3.5	1.3	0.2	0	0
13	4.9	3.1	1.5	0.1	0	0
14	5.1	3.8	1.5	0.3	0	0
15	6.3	3.3	4.7	1.6	1	1
16	6.5	3.0	5.8	2.2	2	2
17	5.6	2.5	3.9	1.1	1	1
18	5.7	2.8	4.5	1.3	1	1
19	6.4	2.8	5.6	2.2	2	2
20	4.7	3.2	1.6	0.2	0	0
21	6.1	3.0	4.9	1.8	2	2
22	5.0	3.4	1.6	0.4	0	0
23	6.4	2.8	5.6	2.1	2	2
24	7.9	3.8	6.4	2.0	2	2
25	6.7	3.0	5.2	2.3	2	2
26	6.7	2.5	5.8	1.8	2	2
27	6.8	3.2	5.9	2.3	2	2
28	4.8	3.0	1.4	0.3	0	0
29	4.8	3.1	1.6	0.2	0	0

In [10]:

```python
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Hitung akurasi
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi:", accuracy)
```

```
Akurasi: 1.0
```

In [11]:
```
# Cetak laporan klasifikasi
print("Laporan Klasifikasi:")
print(classification_report(y_test, y_pred))
```

```
Laporan Klasifikasi:
              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
```

In [12]:
```
# Cetak matriks konfusi
print("Matriks Konfusi:")
print(confusion_matrix(y_test, y_pred))
```

```
Matriks Konfusi:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [13]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.datasets import load_boston
```

In [14]:

```
# Load dataset
boston = load_boston()
df = pd.DataFrame(boston.data, columns=boston.feature_names)
df['PRICE'] = boston.target

# Split dataset menjadi training dan testing
X = df.drop('PRICE', axis=1)
y = df['PRICE']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Buat model Random Forest Regresi
rf = RandomForestRegressor(n_estimators=100, random_state=42)

# Latih model
rf.fit(X_train, y_train)

# Prediksi nilai pada dataset testing
y_pred = rf.predict(X_test)
```

In [15]:

```
# Evaluasi model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("MSE:", mse)
print("R2:", r2)
```

```
MSE: 7.901513892156864
R2: 0.8922527442109116
```

In [16]:

```python
# Buat dataframe untuk hasil prediksi

df_pred = pd.DataFrame({
    'Actual': y_test,
    'Predicted': y_pred
}, index=X_test.index)

df_pred = pd.concat([X_test, df_pred], axis=1)


# Cetak dataframe hasil prediksi
print(df_pred)
```

```
         CRIM    ZN  INDUS  CHAS    NOX     RM    AGE     DIS   RAD    TAX  \
173   0.09178   0.0   4.05   0.0  0.510  6.416   84.1  2.6463   5.0  296.0
274   0.05644  40.0   6.41   1.0  0.447  6.758   32.9  4.0776   4.0  254.0
491   0.10574   0.0  27.74   0.0  0.609  5.983   98.8  1.8681   4.0  711.0
72    0.09164   0.0  10.81   0.0  0.413  6.065    7.8  5.2873   4.0  305.0
452   5.09017   0.0  18.10   0.0  0.713  6.297   91.8  2.3682  24.0  666.0
..        ...   ...    ...   ...    ...    ...    ...     ...   ...    ...
412  18.81100   0.0  18.10   0.0  0.597  4.628  100.0  1.5539  24.0  666.0
436  14.42080   0.0  18.10   0.0  0.740  6.461   93.3  2.0026  24.0  666.0
411  14.05070   0.0  18.10   0.0  0.597  6.657  100.0  1.5275  24.0  666.0
86    0.05188   0.0   4.49   0.0  0.449  6.015   45.1  4.4272   3.0  247.0
75    0.09512   0.0  12.83   0.0  0.437  6.286   45.0  4.5026   5.0  398.0

     PTRATIO       B  LSTAT  Actual  Predicted
173     16.6  395.50   9.04    23.6     22.839
274     17.6  396.90   3.53    32.4     30.676
491     20.1  390.11  18.07    13.6     16.317
72      19.2  390.91   5.52    22.8     23.510
452     20.2  385.09  17.27    16.1     16.819
..       ...     ...    ...     ...        ...
412     20.2   28.79  34.37    17.9     12.790
436     20.2   27.49  18.05     9.6     12.726
411     20.2   35.05  21.22    17.2     13.119
86      18.5  395.99  12.86    22.5     20.603
75      18.7  383.23   8.94    21.4     23.902

[102 rows x 15 columns]
```