```
#Program-9
"""
Write a program to demonstrate Random Forest for classification task
on a given dataset.(Iris Dataset)
"""

# load the iris dataset
from sklearn.datasets import load_iris

iris = load_iris()

# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target

# Count the number of samples
num_samples = X.shape[0]  # The number of rows represents the number
of samples

print(f'Number of samples in the Iris dataset: {num_samples}')
```

```
Number of samples in the Iris dataset: 150
```

```
# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1)

# Count the number of samples in the training and testing sets
train_samples = X_train.shape[0]  # Number of rows in X_train
test_samples = X_test.shape[0]    # Number of rows in X_test

print(f'Number of samples in the training set: {train_samples}')
print(f'Number of samples in the testing set: {test_samples}')
```

```
Number of samples in the training set: 105
Number of samples in the testing set: 45
```

```
# importing random forest classifier from assemble module
from sklearn.ensemble import RandomForestClassifier
# creating a RF classifier
rf = RandomForestClassifier(n_estimators = 100)

# Training the model on the training dataset
# fit function is used to train the model using the training sets as
parameters
rf.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```
# performing predictions on the test dataset
y_pred = rf.predict(X_test)
```

```
# comparing actual response values (y_test) with predicted response
values (y_pred)
from sklearn import metrics
print("Random Forest model accuracy(in %):",
metrics.accuracy_score(y_test, y_pred)*100)
```

```
Random Forest model accuracy(in %): 95.55555555555556
```

```
# Print the actual and predicted values
print("Actual values:", y_test)
print("Predicted values:", y_pred)
```

```
Actual values: [0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 1 0 2 1 0
0 1 2 1 2 1 2 2 0 1
 0 1 2 2 0 2 2 1]
Predicted values: [0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 2 0 2 1
0 0 1 2 1 2 1 2 2 0 1
 0 1 2 2 0 1 2 1]
```

```
import pandas as pd
# Create a DataFrame to display actual and predicted values
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

# Print the table
print(df)
```

```
    Actual  Predicted
0        0          0
1        1          1
2        1          1
3        0          0
4        2          2
5        1          1
6        2          2
7        0          0
8        0          0
9        2          2
10       1          1
11       0          0
12       2          2
13       1          1
14       1          1
15       0          0
16       1          1
17       1          1
18       0          0
19       0          0
20       1          1
21       1          1
22       1          2
```

```
23        0           0
24        2           2
25        1           1
26        0           0
27        0           0
28        1           1
29        2           2
30        1           1
31        2           2
32        1           1
33        2           2
34        2           2
35        0           0
36        1           1
37        0           0
38        1           1
39        2           2
40        2           2
41        0           0
42        2           1
43        2           2
44        1           1
```

```
# Assuming the classes are as follows:
label_mapping = {0: "iris-setosa", 1: "iris-versicolor", 2: "iris-
virginica"}

y_pred=rf.predict([[3, 3, 2, 2]])
print("Result is:", label_mapping[y_pred[0]])
```

```
Result is: iris-setosa
```