```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.datasets import make_classification

# Step 1: Load and explore the Iris dataset
def load_and_train_iris():
    iris = load_iris()
    X, y = iris.data, iris.target

    # Split 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, stratify=y)

    # Normalize features
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Train a RandomForest Classifier
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

    # Predict and evaluate
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    print("Iris Dataset Classification Report:")
    print(classification_report(y_test, y_pred))
    print(f"Accuracy on Iris dataset: {accuracy:.4f}")

    return accuracy

# Step 2: Generate a new simulated dataset
def generate_and_train_simulated():
    X, y = make_classification(n_samples=150, n_features=4, n_informative=3, n_redundant=1, n_classes=3, random_state=42)

    # Split 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, stratify=y)

    # Normalize features
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Train a RandomForest Classifier
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

    # Predict and evaluate
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    print("\nSimulated Dataset Classification Report:")
    print(classification_report(y_test, y_pred))
    print(f"Accuracy on Simulated dataset: {accuracy:.4f}")

    return accuracy

# Run both studies
iris_accuracy = load_and_train_iris()
simulated_accuracy = generate_and_train_simulated()

# Summary Report
print("\nSummary:")
print(f"Accuracy on Iris dataset: {iris_accuracy:.4f}")
print(f"Accuracy on Simulated dataset: {simulated_accuracy:.4f}")
```

```
Iris Dataset Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       0.82      0.90      0.86        10
           2       0.89      0.80      0.84        10

    accuracy                           0.90        30
   macro avg       0.90      0.90      0.90        30
weighted avg       0.90      0.90      0.90        30

Accuracy on Iris dataset: 0.9000

Simulated Dataset Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.70      0.82        10
           1       0.91      1.00      0.95        10
           2       0.83      1.00      0.91        10

    accuracy                           0.90        30
   macro avg       0.91      0.90      0.90        30
weighted avg       0.91      0.90      0.90        30

Accuracy on Simulated dataset: 0.9000

Summary:
Accuracy on Iris dataset: 0.9000
Accuracy on Simulated dataset: 0.9000
```