Start coding or generate with AI.

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

# Sample dataset (replace with real data)
data = {
    'Age': [25, 35, 45, 55, 32, 40, 60, 50, 33, 28],
    'Income': [30000, 60000, 80000, 40000, 70000, 50000, 100000, 90000, 75000, 45000],
    'Credit_History': [1, 0, 1, 0, 1, 1, 0, 1, 1, 0],  # 1 = Good, 0 = Bad
    'Loan_Amount': [5000, 20000, 15000, 10000, 25000, 18000, 30000, 22000, 15000, 12000],
    'Default': [0, 0, 0, 1, 0, 0, 1, 0, 0, 1]  # 0 = No default, 1 = Defaulted
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Define features (X) and target (y)
X = df.drop('Default', axis=1)  # All columns except 'Default'
y = df['Default']  # The target variable

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features (important for many classification algorithms)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

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

# Fit the model on the training data
model.fit(X_train_scaled, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test_scaled)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Display confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n{conf_matrix}')

# Classification report (precision, recall, f1-score)
print(f'Classification Report:\n{classification_report(y_test, y_pred)}')

# Plot the confusion matrix
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["No Default", "Default"], y
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 50.00%
Confusion Matrix:
[[1 1]
 [0 0]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         2
           1       0.00      0.00      0.00         0

    accuracy                           0.50         2
   macro avg       0.50      0.25      0.33         2
weighted avg       1.00      0.50      0.67         2

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```