```
#  Use the Heart Disease database:
#  1. Perform classification with 50%-50% training and testing before data normalization.
#  2. Apply the Min-Max and Z-score normalization techniques, and then perform clas
# sification with 50%-50% training and testing using the SVM classifier.

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Load the dataset
dataset_path = r"C:/Users/HP\OneDrive/Desktop/ml 7th sem codes/datasets/heart.csv"
df = pd.read_csv(dataset_path)

# Explore the dataset
print("Dataset Info:")
df.info()
print("\nShape:", df.shape)
print("\nMissing Values:\n", df.isnull().sum())

# Features and target
X = df.drop('target', axis=1)
y = df['target']

# Split the data into 50% training and 50% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=40)

# Task 1: Classification before normalization
print("\nTask 1: Classification before normalization")
```

```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(report)

# Task 2: Min-Max Normalization
print("\nTask 2: Min-Max Normalization")
min_max_scaler = MinMaxScaler()
X_train_minmax = min_max_scaler.fit_transform(X_train)
X_test_minmax = min_max_scaler.transform(X_test)

svm_model_minmax = SVC(random_state=42)
svm_model_minmax.fit(X_train_minmax, y_train)
y_pred_minmax = svm_model_minmax.predict(X_test_minmax)
accuracy_minmax = accuracy_score(y_test, y_pred_minmax)
report_minmax = classification_report(y_test, y_pred_minmax)
print(f"Accuracy: {accuracy_minmax:.2f}")
print("Classification Report:")
print(report_minmax)

# Task 3: Z-score Normalization
print("\nTask 3: Z-score Normalization")
standard_scaler = StandardScaler()
X_train_standard = standard_scaler.fit_transform(X_train)
X_test_standard = standard_scaler.transform(X_test)

svm_model_standard = SVC(random_state=42)
```

```
svm_model_standard.fit(X_train_standard, y_train)
y_pred_standard = svm_model_standard.predict(X_test_standard)
accuracy_standard = accuracy_score(y_test, y_pred_standard)
report_standard = classification_report(y_test, y_pred_standard)
print(f"Accuracy: {accuracy_standard:.2f}")
print("Classification Report:")
print(report_standard)
```