**Ex 04**  
  
import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Load the dataset

data = pd.read\_csv('D:\\AML\\titanic dataset.csv')

# Display the first few rows of the dataset

print("First few rows of the dataset:")

print(data.head())

# Display information about the dataset

print("\nDataset info:")

print(data.info())

# Check for missing values

print("\nMissing values:")

print(data.isnull().sum())

# Handle missing values

data = data.drop(['Cabin'], axis=1) # Drop 'Cabin' due to too many missing values

data = data.dropna() # Drop other rows with missing values

# Encode categorical variables

le = LabelEncoder()

for column in data.select\_dtypes(include=['object']).columns:

data[column] = le.fit\_transform(data[column])

# Define features and target variable

target\_column = 'Survived'

features = data.drop(target\_column, axis=1)

target = data[target\_column]

# Normalize/scale the features

scaler = StandardScaler()

features\_scaled = scaler.fit\_transform(features)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features\_scaled, target, test\_size=0.2, random\_state=42)

# Initialize and train the Decision Tree Classifier

dt\_model = DecisionTreeClassifier(random\_state=42)

dt\_model.fit(X\_train, y\_train)

# Predict and evaluate the Decision Tree Classifier

y\_pred\_dt = dt\_model.predict(X\_test)

accuracy\_dt = accuracy\_score(y\_test, y\_pred\_dt)

conf\_matrix\_dt = confusion\_matrix(y\_test, y\_pred\_dt)

class\_report\_dt = classification\_report(y\_test, y\_pred\_dt)

print("\nDecision Tree Classifier:")

print("\nAccuracy:", accuracy\_dt)

print("\nConfusion Matrix:\n", conf\_matrix\_dt)

print("\nClassification Report:\n", class\_report\_dt)

# Initialize and train the Random Forest Classifier

rf\_model = RandomForestClassifier(random\_state=42)

rf\_model.fit(X\_train, y\_train)

# Predict and evaluate the Random Forest Classifier

y\_pred\_rf = rf\_model.predict(X\_test)

accuracy\_rf = accuracy\_score(y\_test, y\_pred\_rf)

conf\_matrix\_rf = confusion\_matrix(y\_test, y\_pred\_rf)

class\_report\_rf = classification\_report(y\_test, y\_pred\_rf)

print("\nRandom Forest Classifier:")

print("\nAccuracy:", accuracy\_rf)

print("\nConfusion Matrix:\n", conf\_matrix\_rf)

print("\nClassification Report:\n", class\_report\_rf)

# Compare the performances

print("\nComparison of Models:")

print(f"Decision Tree Accuracy: {accuracy\_dt}")

print(f"Random Forest Accuracy: {accuracy\_rf}")

**Output:**

