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, classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Step 1: Load the dataset

# Replace the path below with the path to your healthcare dataset (CSV file)

data = pd.read\_csv('healthcare\_data.csv')

# Step 2: Inspect the data

print(data.head()) # Display first few rows of the dataset

print(data.info()) # Information about the dataset, null values, data types

# Step 3: Preprocess the data

# Let's assume the last column 'Disease' is the target variable

X = data.drop(columns=['Disease']) # Features (all columns except the target)

y = data['Disease'] # Target variable

# Step 4: Handle missing values (if any)

X.fillna(X.mean(), inplace=True) # Impute missing values with the mean

# Step 5: Split the dataset 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)

# Step 6: Train a classification model (Random Forest Classifier in this case)

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Step 7: Make predictions

y\_pred = model.predict(X\_test)

# Step 8: Evaluate the model

# Accuracy score

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

print(f'Accuracy: {accuracy:.4f}')

# Classification report for precision, recall, f1-score, etc.

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))

# Confusion Matrix

print("Confusion Matrix:")

cm = confusion\_matrix(y\_test, y\_pred)

# Plot the confusion matrix

plt.figure(figsize=(6, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No Disease', 'Disease'], yticklabels=['No Disease', 'Disease'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# Step 9: Feature Importance (optional, to understand what features contribute most)

importances = model.feature\_importances\_

features = X.columns

# Plot feature importance

plt.figure(figsize=(10, 6))

sns.barplot(x=importances, y=features)

plt.title('Feature Importance')

plt.show()