1.Data preprocessing

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

# Load the dataset

data = pd.read\_csv("patient\_data.csv")

# Clean the data (e.g., handle missing values)

data.fillna(data.mean(), inplace=True)

# Select relevant features

features = data[['Age', 'BMI', 'BloodPressure', 'GlucoseLevel']]

target = data['DiseaseStatus'] # Assume 1 for Disease, 0 for No Disease

# Split the data into training and testing sets

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

# Scale the data

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Save preprocessed data (optional)

# pd.to\_pickle(X\_train, 'X\_train.pkl')

# pd.to\_pickle(X\_test, 'X\_test.pkl')from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

# Load the dataset

data = pd.read\_csv("patient\_data.csv")

# Clean the data (e.g., handle missing values)

data.fillna(data.mean(), inplace=True)

# Select relevant features

features = data[['Age', 'BMI', 'BloodPressure', 'GlucoseLevel']]

target = data['DiseaseStatus'] # Assume 1 for Disease, 0 for No Disease

# Split the data into training and testing sets

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

# Scale the data

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Save preprocessed data (optional)

# pd.to\_pickle(X\_train, 'X\_train.pkl')

# pd.to\_pickle(X\_test, 'X\_test.pkl')

2.Model training

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, f1\_score, roc\_auc\_score

import pickle

# Load the preprocessed data

X\_train = pd.read\_pickle('X\_train.pkl')

X\_test = pd.read\_pickle('X\_test.pkl')

y\_train = pd.read\_pickle('y\_train.pkl')

y\_test = pd.read\_pickle('y\_test.pkl')

# Train a Random Forest Classifier

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

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

# Predict on test data

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

# Evaluate the model

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

f1 = f1\_score(y\_test, y\_pred)

roc\_auc = roc\_auc\_score(y\_test, model.predict\_proba(X\_test)[:, 1])

print(f"Accuracy: {accuracy}")

print(f"F1-score: {f1}")

print(f"ROC-AUC: {roc\_auc}")

# Save the trained model

with open('model.pkl', 'wb') as f:

pickle.dump(model, f)

3.Streamlit app

import streamlit as st

import pickle

import numpy as np

# Load the trained model

model = pickle.load(open("model.pkl", "rb"))

# Streamlit app

st.title("AI-powered Disease Prediction")

st.write("Enter patient data to predict disease status.")

# User inputs

age = st.number\_input("Age", min\_value=1, max\_value=120)

bmi = st.number\_input("BMI", min\_value=10.0, max\_value=50.0)

blood\_pressure = st.number\_input("Blood Pressure", min\_value=80, max\_value=180)

glucose\_level = st.number\_input("Glucose Level", min\_value=50, max\_value=200)

# Predict button

if st.button("Predict"):

input\_data = np.array([[age, bmi, blood\_pressure, glucose\_level]])

prediction = model.predict(input\_data)

if prediction[0] == 1:

st.write("Prediction: \*\*Disease\*\*")

else:

st.write("Prediction: \*\*No Disease\*\*")

4.Requirements file

streamlit==1.11.1

scikit-learn==1.0.2

pandas==1.3.3

numpy==1.21.2

matplotlib==3.4.3

seaborn==0.11.2

5.Model evaluation

import pickle

from sklearn.metrics import accuracy\_score, f1\_score, roc\_auc\_score, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Load the trained model

model = pickle.load(open("model.pkl", "rb"))

# Load the test data

X\_test = pd.read\_pickle('X\_test.pkl')

y\_test = pd.read\_pickle('y\_test.pkl')

# Predict on the test data

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

# Evaluate the model

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

f1 = f1\_score(y\_test, y\_pred)

roc\_auc = roc\_auc\_score(y\_test, model.predict\_proba(X\_test)[:, 1])

print(f"Accuracy: {accuracy}")

print(f"F1-score: {f1}")

print(f"ROC-AUC: {roc\_auc}")

# Confusion Matrix

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

# Plot confusion matrix

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

plt.title("Confusion Matrix")

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.show()

6.Deployment script

#!/bin/bash

# Deploy the Streamlit app to Streamlit Cloud

streamlit run app.py