Smart Healthcare Data Analyzer for Disease Pattern Detection

```
# Smart Healthcare Data Analyzer for Disease Pattern Detection
import streamlit as st
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

st.title("■ Smart Healthcare Data Analyzer for Disease Pattern Detection")

uploaded_file = st.sidebar.file_uploader("Upload CSV file", type=["csv"])
if uploaded_file is None:
    st.sidebar.info("No file uploaded. Using demo data.")
    np.random.seed(42)
    n = 500
    df = pd.DataFrame({
        'age': np.random.randint(18, 90, n),
        'bmi': np.round(np.random.normal(27, 5, n), 1),
        'blood_pressure': np.random.randint(80, 160, n),
        'cholesterol': np.random.randint(150, 300, n),
        'smoker': np.random.choice([0,1], n, p=[0.8,0.2]),
        'physical_activity': np.random.randint(0,5,n),
        'gender': np.random.choice(['M','F'], n),
    })
    df['disease'] = ((df['bmi']>30).astype(int) + (df['blood_pressure']>140).astype(int) + df['smoke
    df['disease'] = (df['disease']>=2).astype(int)
else:
    df = pd.read_csv(uploaded_file)

# EDA and preprocessing steps
st.dataframe(df.head())
sns.heatmap(df.corr(), annot=False)

# Clustering and PCA
cluster_features = df.select_dtypes(include=[np.number])
kmeans = KMeans(n_clusters=3, random_state=42).fit(cluster_features)
df['cluster'] = kmeans.labels_

# Predictive modeling
target = 'disease'
if target in df.columns:
    X = df.drop(columns=[target])
    y = df[target]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    acc = accuracy_score(y_test, preds)
    st.write(f"Accuracy: {acc:.2f}")
```