```
# -*- coding: utf-8 -*-
"""Copy of sp NM pro.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1lD8YbdeoxSn0Lw1hy8xLNVzGywnCxyu4
"""

!pip install gradio
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
try:
    df = pd.read_csv('/content/healthBook1.csv')
    print("Dataset loaded successfully.")
    print(df.head())
    print(df.info())
except FileNotFoundError:
    print("Error: The file '/content/healthBook1.csv' was not found.")
    exit()

# Data Preprocessing.
df = df.dropna()
print("\nMissing values handled (rows with NaNs dropped).")

# Encode categorical features
# Identify categorical columns
categorical_cols = [col for col in df.columns if df[col].dtype == 'object']
print("\nCategorical columns:", categorical_cols)

label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
```

```
    label_encoders[col] = le
print("\nCategorical columns encoded.")
print(df.head())

# Separate features (X) and target (y)
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
print("\nFeatures (X) shape:", X.shape)
print("Target (y) shape:", y.shape)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
3, random_state=42)
print("\nData split into training and testing sets.")
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)

# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
print("\nFeatures scaled using StandardScaler.")

# Model Training
model = LogisticRegression(random_state=42)
model.fit(X_train, y_train)
print("\nLogistic Regression model trained.")

# Model Evaluation
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy of the Logistic Regression model: {accuracy:.4f}")

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Create confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
```

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```