```
# Step 1: Install required libraries
!pip install pandas scikit-learn matplotlib seaborn --quiet

# Step 2: Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Step 3: Load the dataset
from google.colab import files
uploaded = files.upload()

# Load into DataFrame
filename = next(iter(uploaded))
df = pd.read_csv(filename)

# Step 4: Inspect columns
print("Columns in the dataset:\n", df.columns)

# Guess target column (assume last column is disease if not explicitly named)
target_col = df.columns[-1]
print(f"\nAssumed target column: {target_col}")

# Step 5: Data cleaning
print("\nMissing values:\n", df.isnull().sum())
df.dropna(inplace=True)  # Remove rows with missing data

# Step 6: Encode categorical features
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Step 7: Feature/Target split
X = df.drop(target_col, axis=1)
y = df[target_col]

# Step 8: Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 9: Train/test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Step 10: Train model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 11: Evaluate model
y_pred = model.predict(X_test)
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
# Confusion matrix
plt.figure(figsize=(10, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Step 12: Example prediction
sample_input = X.iloc[0:1].values  # use first row as a sample
pred = model.predict(scaler.transform(sample_input))
print("Sample prediction (encoded):", pred)
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving ai_healthcare_disease_prediction_100k.csv to ai_healthcare_disease_prediction_100k (1).
Columns in the dataset:
 Index(['Patient_ID', 'Age', 'Gender', 'Medical_History', 'Diagnosis',
       'Treatment', 'Lab_Glucose_mg/dL', 'Lab_Cholesterol_mg/dL',
       'Imaging_Summary', 'Heart_Rate_bpm', 'Steps_Per_Day', 'Activity_Level',
       'Region'],
      dtype='object')

Assumed target column: Region

Missing values:
 Patient_ID                  0
Age                          0
Gender                       0
Medical_History          20033
Diagnosis                    0
Treatment                19960
Lab_Glucose_mg/dL            0
Lab_Cholesterol_mg/dL        0
Imaging_Summary              0
Heart_Rate_bpm               0
Steps_Per_Day                0
Activity_Level               0
Region                       0
dtype: int64
Accuracy Score: 0.20713895180816996

Classification Report:
               precision    recall  f1-score   support

           0       0.21      0.26      0.23      2583
           1       0.19      0.21      0.20      2509
           2       0.21      0.21      0.21      2637
           3       0.21      0.20      0.21      2522
           4       0.20      0.16      0.18      2552

    accuracy                           0.21     12803
   macro avg       0.21      0.21      0.21     12803
weighted avg       0.21      0.21      0.21     12803
```