XGBoost Classification for Heart Disease Prediction (Without Scaling)

This notebook builds an XGBoost classifier to predict heart disease using the synthetic dataset. We will reduce to the top 7 features based on importance, without scaling.

Step 1: Import Libraries

We import necessary libraries for data handling, preprocessing, modeling, and evaluation.

In [10]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import xgboost as xgb
import pickle
```

Step 2: Load the Dataset

Load the CSV file into a pandas DataFrame and display basic information.

In [11]:
```
df = pd.read_csv('synthetic_heart_disease_dataset.csv')
print("Dataset shape:", df.shape)
print("Columns:", df.columns.tolist())
print("First 5 rows:")
print(df.head())
print("\nData types:")
print(df.dtypes)
print("\nMissing values:")
print(df.isnull().sum())
```

```
Dataset shape: (50000, 21)
Columns: ['Age', 'Gender', 'Weight', 'Height', 'BMI', 'Smoking', 'Alcohol_Intake',
'Physical_Activity', 'Diet', 'Stress_Level', 'Hypertension', 'Diabetes', 'Hyperlipid
emia', 'Family_History', 'Previous_Heart_Attack', 'Systolic_BP', 'Diastolic_BP', 'He
art_Rate', 'Blood_Sugar_Fasting', 'Cholesterol_Total', 'Heart_Disease']
First 5 rows:
   Age  Gender  Weight  Height   BMI  Smoking Alcohol_Intake  \
0   48    Male      78     157  26.4    Never            NaN
1   35  Female      73     163  33.0    Never            Low
2   79  Female      88     152  32.3    Never            NaN
3   75    Male     106     171  37.4    Never       Moderate
4   34  Female      65     191  18.5  Current            NaN

  Physical_Activity     Diet Stress_Level  ...  Diabetes  Hyperlipidemia  \
0         Sedentary  Healthy       Medium  ...         0               1
1            Active  Average         High  ...         0               1
2          Moderate  Average       Medium  ...         0               0
3          Moderate  Average          Low  ...         0               1
4         Sedentary  Healthy          Low  ...         1               0

   Family_History  Previous_Heart_Attack  Systolic_BP  Diastolic_BP  \
0               1                      0          104            99
1               1                      0          111            72
2               1                      0          116           102
3               0                      0          171            92
4               0                      0          164            67

   Heart_Rate  Blood_Sugar_Fasting  Cholesterol_Total  Heart_Disease
0          71                  165                200              0
1          60                  145                206              0
2          78                  148                208              0
3         109                  105                290              1
4         108                  116                220              1

[5 rows x 21 columns]

Data types:
Age                        int64
Gender                    object
Weight                     int64
Height                     int64
BMI                      float64
Smoking                   object
Alcohol_Intake            object
Physical_Activity         object
Diet                      object
Stress_Level              object
Hypertension               int64
Diabetes                   int64
Hyperlipidemia             int64
Family_History             int64
Previous_Heart_Attack      int64
Systolic_BP                int64
Diastolic_BP               int64
Heart_Rate                 int64
Blood_Sugar_Fasting        int64
```

```
Cholesterol_Total          int64
Heart_Disease              int64
dtype: object

Missing values:
Age                         0
Gender                      0
Weight                      0
Height                      0
BMI                         0
Smoking                     0
Alcohol_Intake          20109
Physical_Activity           0
Diet                        0
Stress_Level                0
Hypertension                0
Diabetes                    0
Hyperlipidemia              0
Family_History              0
Previous_Heart_Attack       0
Systolic_BP                 0
Diastolic_BP                0
Heart_Rate                  0
Blood_Sugar_Fasting         0
Cholesterol_Total           0
Heart_Disease               0
dtype: int64
```

Step 3: Preprocess the Data

Handle missing values by filling with mode for categorical, encode categorical variables to numerical.

In [12]:
```python
# Handle missing values
df['Alcohol_Intake'].fillna(df['Alcohol_Intake'].mode()[0], inplace=True)

# Encode categorical variables
categorical_cols = ['Gender', 'Smoking', 'Alcohol_Intake', 'Physical_Activity', 'Di
le = LabelEncoder()
for col in categorical_cols:
    df[col] = le.fit_transform(df[col])

print("After preprocessing:")
print(df.head())
```

```
After preprocessing:
   Age  Gender  Weight  Height   BMI  Smoking  Alcohol_Intake  \
0   48       1      78     157  26.4        2               1   
1   35       0      73     163  33.0        2               1   
2   79       0      88     152  32.3        2               1   
3   75       1     106     171  37.4        2               2   
4   34       0      65     191  18.5        0               1   

   Physical_Activity  Diet  Stress_Level  ...  Diabetes  Hyperlipidemia  \
0                  2     1             2  ...         0               1   
1                  0     0             0  ...         0               1   
2                  1     0             2  ...         0               0   
3                  1     0             1  ...         0               1   
4                  2     1             1  ...         1               0   

   Family_History  Previous_Heart_Attack  Systolic_BP  Diastolic_BP  \
0               1                      0          104            99   
1               1                      0          111            72   
2               1                      0          116           102   
3               0                      0          171            92   
4               0                      0          164            67   

   Heart_Rate  Blood_Sugar_Fasting  Cholesterol_Total  Heart_Disease  
0          71                  165                200              0  
1          60                  145                206              0  
2          78                  148                208              0  
3         109                  105                290              1  
4         108                  116                220              1  

[5 rows x 21 columns]
```

```
C:\Users\yahya\AppData\Local\Temp\ipykernel_24524\2640066891.py:2: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['Alcohol_Intake'].fillna(df['Alcohol_Intake'].mode()[0], inplace=True)
```

Step 4: Feature Selection Function

Define a function to train an initial XGBoost model and select the top 7 features based on gain importance.

In [21]:
```
def feature_importance_function(X, y, n_features=8):
    """
    Computes feature importance using XGBoost and returns the top n features.
    """
    model = xgb.XGBClassifier(objective='binary:logistic', n_estimators=100, random
    model.fit(X, y)
```

```
    importance = model.get_booster().get_score(importance_type='gain')
    sorted_features = sorted(importance.items(), key=lambda x: x[1], reverse=True)
    print("Top", n_features, "features by importance:")
    for i, (feature, score) in enumerate(sorted_features[:n_features]):
        print(f"{i+1}. {feature}: {score:.4f}")
    top_features = [f[0] for f in sorted_features[:n_features]]
    return top_features

X = df.drop('Heart_Disease', axis=1)
y = df['Heart_Disease']
top_features = feature_importance_function(X, y, 8)
X_selected = X[top_features]
```

```
Top 8 features by importance:
1. Hypertension: 334.2283
2. Cholesterol_Total: 271.0656
3. Diabetes: 248.6891
4. Age: 230.3551
5. Previous_Heart_Attack: 126.4948
6. Systolic_BP: 0.0023
7. Heart_Rate: 0.0007
8. Diastolic_BP: 0.0001
```

Step 5: Split the Data

Split the selected features and target into training and testing sets.

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2, r
print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```

```
Training set shape: (40000, 7)
Testing set shape: (10000, 7)
```

Step 6: Train the XGBoost Model

Initialize and train the XGBoost classifier on the training data.

In [15]:
```
model = xgb.XGBClassifier(
    objective='binary:logistic',
    n_estimators=100,
    max_depth=6,
    learning_rate=0.1,
    random_state=42
)
model.fit(X_train, y_train)
print("Model trained.")
```

```
Model trained.
```

Step 7: Evaluate the Model

Make predictions on the test set and evaluate performance.

In [16]:
```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 1.0000

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5365
           1       1.00      1.00      1.00      4635

    accuracy                           1.00     10000
   macro avg       1.00      1.00      1.00     10000
weighted avg       1.00      1.00      1.00     10000


Confusion Matrix:
[[5365    0]
 [   0 4635]]
```

Step 8: Save the Model

Save the trained model to a pickle file.

In [17]:
```
with open('xgboost_without_scaler.pkl', 'wb') as f:
    pickle.dump(model, f)
print("Model saved to xgboost_without_scaler.pkl")
```

```
Model saved to xgboost_without_scaler.pkl
```