Predicting Heart Attack Likelihood in Youths of India using Machine Learning Techniques

1. Introduction

Heart disease is a leading cause of morbidity and mortality globally. Early detection of heart attack risk in young individuals can help in preventive healthcare measures. This project aims to develop a machine learning model to predict the likelihood of heart attacks in Indian youths based on demographic, lifestyle, and clinical parameters.

2. Dataset Overview

The dataset consists of multiple attributes, including:

- **Demographics**: Age, Gender, Region, Urban/Rural Status, Socioeconomic Status (SES)
- **Lifestyle Factors**: Smoking status, Alcohol consumption, Diet type, Physical activity level, Screen time, and Sleep duration
- **Clinical Factors**: Blood pressure, Heart rate, ECG results, Cholesterol levels, Blood oxygen levels, and Triglyceride levels
- **Health History**: Family history of heart disease, Diabetes, Hypertension, and Stress levels
- **Target Variable**: Heart Attack Likelihood (Yes/No)

3. Aim of the Project

The primary objective is to:

1. Identify key factors influencing heart attack likelihood.
2. Develop an accurate predictive model using machine learning techniques.
3. Compare different classification algorithms to determine the most effective approach.

4. Data Preprocessing

To ensure high data quality, the following preprocessing steps were performed:

- **Handling Missing Values**: K-Nearest Neighbors (KNN) Imputation was used for numerical attributes, while forward and backward filling was applied to categorical attributes.
- **Feature Engineering**: Blood pressure values were split into systolic and diastolic components.
- **Balancing the Dataset**: Resampling techniques (upsampling minority class and downsampling majority class) were used to address class imbalance.
- **Encoding Categorical Variables**: Ordinal encoding and label encoding were applied to categorical features.
- **Feature Scaling**: Standardization (using StandardScaler) was performed to normalize numerical attributes.

5. Machine Learning Techniques Used

Several classification algorithms were applied:

- **Decision Tree Classifier**: A tree-based approach using Gini impurity and entropy for splitting.
- **Random Forest Classifier**: An ensemble method aggregating multiple decision trees.
- **Support Vector Machine (SVM)**: Utilized with an RBF kernel to capture non-linear relationships.
- **K-Nearest Neighbors (KNN)**: A distance-based method for classification.
- **Naïve Bayes**: A probabilistic classifier based on Bayes' theorem.
- **Boosting Algorithms**:
    - **AdaBoost**: Sequentially improves weak classifiers by adjusting weights.
    - **Gradient Boosting**: Uses gradient descent to optimize weak learners.
    - **XGBoost**: An optimized gradient boosting algorithm for better performance.
- **Bagging Algorithm**:

    - **Bootstrap Aggregating (Bagging)**: Used with Random Forest to improve stability and accuracy.
- **Hyperparameter Tuning**:
    - **Grid Search and Random Search**: Applied to optimize parameters for Random Forest, SVM, and Boosting models.
- **Evaluation Metrics**:
    - Confusion Matrix, Accuracy, Precision, Recall, F1-score, and ROC-AUC Curve.

6. Model Evaluation & Insights

- **Performance Metrics**: Accuracy, Precision, Recall, F1-score, and ROC-AUC were computed.
- **Feature Importance**:
    - **Top Predictors**: Diastolic BP (10.8%), Systolic BP (10.6%), Triglyceride Levels (10.3%).
    - **Lifestyle factors (screen time, sleep duration)** had moderate influence.
- **Best Performing Model**:
    - **Random Forest (with tuning)** achieved **92% accuracy** and an **ROC-AUC score of 0.95**.
    - **SVM (with RBF kernel)** reached an **accuracy of 80%** after hyperparameter tuning.

7. Conclusion

This project successfully demonstrated that machine learning can effectively predict heart attack risk based on lifestyle, clinical, and demographic factors. The findings highlight the importance of blood pressure and lipid levels as primary risk indicators. The developed model can be utilized in preventive healthcare initiatives, insurance risk assessment, and public health policies to mitigate heart disease risks among young individuals.

Future work could involve deep learning approaches and real-time monitoring systems for even more accurate predictions.

In [3]:
```
import pandas as pd
from sklearn.impute import KNNImputer
from sklearn.preprocessing import OrdinalEncoder


df = pd.read_csv("demographics.csv")


ordinal_encoder = OrdinalEncoder()
df[["Gender", "Region", "Urban/Rural", "SES"]] = ordinal_encoder.fit_transform(
    df[["Gender", "Region", "Urban/Rural", "SES"]]
)


imputer = KNNImputer(n_neighbors=5, weights="uniform")
df_demographics = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)

df_demographics[["Gender", "Region", "Urban/Rural", "SES"]] = ordinal_encoder.inverse_transform(
    df_demographics[["Gender", "Region", "Urban/Rural", "SES"]]
)


df_demographics.head(10)
```

Out[3]:

	Age	Gender	Region	Urban/Rural	SES
0	30.0	Female	East	Urban	High
1	26.6	Female	North-East	Urban	Low
2	23.2	Female	North	Urban	Low
3	27.0	Female	East	Rural	Low
4	21.0	Female	East	Urban	Low
5	20.0	Male	West	Rural	Low
6	29.0	Male	North	Rural	Middle
7	32.0	Female	North	Urban	Low
8	19.0	Female	West	Rural	Low
9	30.4	Male	West	Urban	High

In [5]:
```
df_lifestyle=pd.read_csv('lifestyle.csv')
df_lifestyle.head(10)
df_lifestyle.isna().sum()
```

Out[5]:
```
Smoking Status               3000
Alcohol Consumption          3000
Diet Type                       0
Physical Activity Level         0
Screen Time (hrs/day)        3000
Sleep Duration (hrs/day)     3000
Dietary Preferences         10000
Physical Activity           10000
dtype: int64
```

In [7]:
```
import pandas as pd
from sklearn.impute import KNNImputer

num_cols = ["Screen Time (hrs/day)", "Sleep Duration (hrs/day)"]
cat_cols = ["Smoking Status", "Alcohol Consumption", "Diet Type", "Physical Activity Level"]
```

```
# KNN Imputation for numerical columns
knn_imputer = KNNImputer(n_neighbors=5)
df_lifestyle[num_cols] = knn_imputer.fit_transform(df_lifestyle[num_cols])

# Backward and forward fill for categorical columns
df_lifestyle[cat_cols] = df_lifestyle[cat_cols].bfill().ffill()

df_lifestyle.head()
```

Out[7]:

	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	Sleep Duration (hrs/day)	Dietary Preferences	Physical Activity
0	Never	Occasionally	Non-Vegetarian	Sedentary	3.0	8.0	NaN	NaN
1	Occasionally	Occasionally	Non-Vegetarian	Sedentary	15.0	9.0	NaN	NaN
2	Occasionally	Never	Vegan	High	15.0	6.2	NaN	NaN
3	Occasionally	Never	Vegetarian	Sedentary	6.0	7.0	NaN	NaN
4	Occasionally	Occasionally	Vegetarian	Moderate	8.4	9.0	NaN	NaN

In [9]:
```
df_lifestyle.drop(columns=["Dietary Preferences", "Physical Activity"], inplace=True)
df_lifestyle.head()
```

Out[9]:

	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	Sleep Duration (hrs/day)
0	Never	Occasionally	Non-Vegetarian	Sedentary	3.0	8.0
1	Occasionally	Occasionally	Non-Vegetarian	Sedentary	15.0	9.0
2	Occasionally	Never	Vegan	High	15.0	6.2
3	Occasionally	Never	Vegetarian	Sedentary	6.0	7.0
4	Occasionally	Occasionally	Vegetarian	Moderate	8.4	9.0

In [11]:
```
df_demographics_lifestyle=pd.concat([df_demographics,df_lifestyle],axis=1)
df_demographics_lifestyle.head()
```

Out[11]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	Sleep Duration (hrs/day)
0	30.0	Female	East	Urban	High	Never	Occasionally	Non-Vegetarian	Sedentary	3.0	8.0
1	26.6	Female	North-East	Urban	Low	Occasionally	Occasionally	Non-Vegetarian	Sedentary	15.0	9.0
2	23.2	Female	North	Urban	Low	Occasionally	Never	Vegan	High	15.0	6.2
3	27.0	Female	East	Rural	Low	Occasionally	Never	Vegetarian	Sedentary	6.0	7.0
4	21.0	Female	East	Urban	Low	Occasionally	Occasionally	Vegetarian	Moderate	8.4	9.0

In [13]:
```
df_clinical=pd.read_csv('clinical.csv')
df_clinical.head()
```

Out[13]:

	Blood Pressure (systolic/diastolic mmHg)	Resting Heart Rate (bpm)	ECG Results	Chest Pain Type	Maximum Heart Rate Achieved	Exercise Induced Angina	Blood Oxygen Levels (SpO2%)	Triglyceride Levels (mg/dL)
0	177.0/63.1	82.0	Normal	NaN	183.0	No	94.1	NaN
1	NaN	76.0	NaN	Non-anginal	NaN	No	97.1	341.0
2	138.3/76.6	NaN	NaN	Typical	164.0	No	92.7	373.0
3	177.1/90.0	106.0	NaN	Non-anginal	NaN	No	98.4	102.0
4	130.7/108.8	73.0	Normal	Atypical	216.0	NaN	94.9	235.0

In [15]:
```
df_clinical[['Systolic BP', 'Diastolic BP']] = df_clinical['Blood Pressure (systolic/diastolic mmHg)'].str.split('/


df_clinical.drop(columns=['Blood Pressure (systolic/diastolic mmHg)'], inplace=True)
df_clinical.head()
```

Out[15]:

	Resting Heart Rate (bpm)	ECG Results	Chest Pain Type	Maximum Heart Rate Achieved	Exercise Induced Angina	Blood Oxygen Levels (SpO2%)	Triglyceride Levels (mg/dL)	Systolic BP	Diastolic BP
0	82.0	Normal	NaN	183.0	No	94.1	NaN	177.0	63.1
1	76.0	NaN	Non-anginal	NaN	No	97.1	341.0	NaN	NaN
2	NaN	NaN	Typical	164.0	No	92.7	373.0	138.3	76.6
3	106.0	NaN	Non-anginal	NaN	No	98.4	102.0	177.1	90.0
4	73.0	Normal	Atypical	216.0	NaN	94.9	235.0	130.7	108.8

In [35]:

```
num_cols = ["Systolic BP", "Diastolic BP", "Resting Heart Rate (bpm)",
            "Maximum Heart Rate Achieved", "Blood Oxygen Levels (SpO2%)",
            "Triglyceride Levels (mg/dL)"]
cat_cols = ["ECG Results", "Chest Pain Type", "Exercise Induced Angina"]

# KNN Imputation for numerical columns
knn_imputer = KNNImputer(n_neighbors=5)
df_clinical[num_cols] = knn_imputer.fit_transform(df_clinical[num_cols])

# Backward and forward fill for categorical columns
df_clinical[cat_cols] = df_clinical[cat_cols].bfill().ffill()
df_clinical.head()
```

Out[35]:

	Resting Heart Rate (bpm)	ECG Results	Chest Pain Type	Maximum Heart Rate Achieved	Exercise Induced Angina	Blood Oxygen Levels (SpO2%)	Triglyceride Levels (mg/dL)	Systolic BP	Diastolic BP
0	82.0	Normal	Non-anginal	183.0	No	94.1	275.2	177.00	63.1
1	76.0	Normal	Non-anginal	153.8	No	97.1	341.0	143.66	94.2
2	97.0	Normal	Typical	164.0	No	92.7	373.0	138.30	76.6
3	106.0	Normal	Non-anginal	159.6	No	98.4	102.0	177.10	90.0
4	73.0	Normal	Atypical	216.0	No	94.9	235.0	130.70	108.8

In [37]:
```
df_clinical.isna().sum()
```

Out[37]:
```
Resting Heart Rate (bpm)       0
ECG Results                    0
Chest Pain Type                0
Maximum Heart Rate Achieved    0
Exercise Induced Angina        0
Blood Oxygen Levels (SpO2%)    0
Triglyceride Levels (mg/dL)    0
Systolic BP                    0
Diastolic BP                   0
dtype: int64
```

In [39]:
```
df_demographics_lifestyle_clinical=pd.concat([df_demographics_lifestyle,df_clinical],axis=1)
df_demographics_lifestyle_clinical.head()
```

Out[39]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	Sleep Duration (hrs/day)	Resting Heart Rate (bpm)	EC Result
0	30.0	Female	East	Urban	High	Never	Occasionally	Non-Vegetarian	Sedentary	3.0	8.0	82.0	Norm
1	26.6	Female	North-East	Urban	Low	Occasionally	Occasionally	Non-Vegetarian	Sedentary	15.0	9.0	76.0	Norm
2	23.2	Female	North	Urban	Low	Occasionally	Never	Vegan	High	15.0	6.2	97.0	Norm
3	27.0	Female	East	Rural	Low	Occasionally	Never	Vegetarian	Sedentary	6.0	7.0	106.0	Norm
4	21.0	Female	East	Urban	Low	Occasionally	Occasionally	Vegetarian	Moderate	8.4	9.0	73.0	Norm

In [41]:
```
df_health=pd.read_csv('health.csv')
df_health.head()
```

Out[41]:

	Family History of Heart Disease	Diabetes	Hypertension	Cholesterol Levels (mg/dL)	BMI (kg/m²)	Stress Level
0	No	NaN	Yes	148.0	34.4	NaN
1	No	No	NaN	NaN	25.0	High
2	Yes	NaN	No	NaN	NaN	Low
3	NaN	NaN	No	137.0	19.0	Medium
4	Yes	NaN	NaN	NaN	28.0	Low

In [43]:
```
num_cols = ["Cholesterol Levels (mg/dL)","BMI (kg/m²)"]
cat_cols = ["Family History of Heart Disease", "Diabetes", "Hypertension","Stress Level"]

# KNN Imputation for numerical columns
knn_imputer = KNNImputer(n_neighbors=5)
df_health[num_cols] = knn_imputer.fit_transform(df_health[num_cols])

# Backward and forward fill for categorical columns
```

```
df_health[cat_cols] = df_health[cat_cols].bfill().ffill()
df_health.head()
```

Out[43]:

	Family History of Heart Disease	Diabetes	Hypertension	Cholesterol Levels (mg/dL)	BMI (kg/m²)	Stress Level
0	No	No	Yes	148.000000	34.40000	High
1	No	No	No	191.000000	25.00000	High
2	Yes	Yes	No	199.576956	27.43412	Low
3	Yes	Yes	No	137.000000	19.00000	Medium
4	Yes	Yes	No	251.400000	28.00000	Low

In [45]:
```
df_demographics_lifestyle_clinical_health=pd.concat([df_demographics_lifestyle_clinical,df_health],axis=1)
df_demographics_lifestyle_clinical_health.head()
```

Out[45]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Blood Oxygen Levels (SpO2%)	Triglyceride Levels (mg/dL)
0	30.0	Female	East	Urban	High	Never	Occasionally	Non-Vegetarian	Sedentary	3.0	...	94.1	275.2
1	26.6	Female	North-East	Urban	Low	Occasionally	Occasionally	Non-Vegetarian	Sedentary	15.0	...	97.1	341.0
2	23.2	Female	North	Urban	Low	Occasionally	Never	Vegan	High	15.0	...	92.7	373.0
3	27.0	Female	East	Rural	Low	Occasionally	Never	Vegetarian	Sedentary	6.0	...	98.4	102.0
4	21.0	Female	East	Urban	Low	Occasionally	Occasionally	Vegetarian	Moderate	8.4	...	94.9	235.0

5 rows × 26 columns

In [47]:
```
df_likelihood=pd.read_csv('heart_attack_likelihood.csv')
df_likelihood.head()
```

Out[47]:

	Heart Attack Likelihood
0	No
1	No
2	Yes
3	NaN
4	No

In [49]:
```
cat_cols = ["Heart Attack Likelihood"]
df_likelihood[cat_cols] = df_likelihood[cat_cols].bfill().ffill()
df_demographics_lifestyle_clinical_health_likelihood=pd.concat([df_demographics_lifestyle_clinical_health,df_likeli
df_demographics_lifestyle_clinical_health_likelihood.head()
```

Out[49]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Systolic BP
0	30.0	Female	East	Urban	High	Never	Occasionally	Non-Vegetarian	Sedentary	3.0	...	275.2	177.00
1	26.6	Female	North-East	Urban	Low	Occasionally	Occasionally	Non-Vegetarian	Sedentary	15.0	...	341.0	143.66
2	23.2	Female	North	Urban	Low	Occasionally	Never	Vegan	High	15.0	...	373.0	138.30
3	27.0	Female	East	Rural	Low	Occasionally	Never	Vegetarian	Sedentary	6.0	...	102.0	177.10
4	21.0	Female	East	Urban	Low	Occasionally	Occasionally	Vegetarian	Moderate	8.4	...	235.0	130.70

5 rows × 27 columns

In [51]:
```
df_demographics_lifestyle_clinical_health_likelihood.to_csv("Final1_Dataset.csv", index=False)
```

In [ ]:
```

```

Random Forest Classifier Algorithmn

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df_final=pd.read_csv('Final1_Dataset.csv')
df_final.head()
```

Out[5]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Systolic BP
0	30.0	Female	East	Urban	High	Never	Occasionally	Non-Vegetarian	Sedentary	3.0	...	275.2	177.00
1	26.6	Female	North-East	Urban	Low	Occasionally	Occasionally	Non-Vegetarian	Sedentary	15.0	...	341.0	143.66
2	23.2	Female	North	Urban	Low	Occasionally	Never	Vegan	High	15.0	...	373.0	138.30
3	27.0	Female	East	Rural	Low	Occasionally	Never	Vegetarian	Sedentary	6.0	...	102.0	177.10
4	21.0	Female	East	Urban	Low	Occasionally	Occasionally	Vegetarian	Moderate	8.4	...	235.0	130.70

5 rows × 27 columns

In [7]:
```
df_final.shape
```

Out[7]: (10000, 27)

In [9]:
```
df_final.isna().sum()
```

Out[9]:
```
Age                                0
Gender                             0
Region                             0
Urban/Rural                        0
SES                                0
Smoking Status                     0
Alcohol Consumption                0
Diet Type                          0
Physical Activity Level            0
Screen Time (hrs/day)              0
Sleep Duration (hrs/day)           0
Resting Heart Rate (bpm)           0
ECG Results                        0
Chest Pain Type                    0
Maximum Heart Rate Achieved        0
Exercise Induced Angina            0
Blood Oxygen Levels (SpO2%)        0
Triglyceride Levels (mg/dL)        0
Systolic BP                        0
Diastolic BP                       0
Family History of Heart Disease    0
Diabetes                           0
Hypertension                       0
Cholesterol Levels (mg/dL)         0
BMI (kg/m²)                        0
Stress Level                       0
Heart Attack Likelihood            0
dtype: int64
```

In [11]:
```
df_final['Heart Attack Likelihood'].value_counts()
```

Out[11]:
```
Heart Attack Likelihood
No     7994
Yes    2006
Name: count, dtype: int64
```

In [13]:
```
df_final_yes=df_final[df_final['Heart Attack Likelihood']=='Yes']
df_final_yes.shape
```

Out[13]:
```
(2006, 27)
```

In [15]:
```
df_final_no=df_final[df_final['Heart Attack Likelihood']=='No']
df_final_no.shape
```

Out[15]: (7994, 27)

In [17]:
```
from sklearn.utils import resample
df_final_yes_up=resample(df_final_yes,replace=True,n_samples=4000,random_state=10)
df_final_yes_up.shape
```

Out[17]: (4000, 27)

In [19]:
```
df_final_no_down=resample(df_final_no,replace=True,n_samples=6000,random_state=10)
df_final_no_down.shape
```

Out[19]: (6000, 27)

In [21]:
```
df_final_new=pd.concat([df_final_yes_up,df_final_no_down])
df_final_new.head(30)
```

Out[21]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Sy
6464	26.8	Female	North-East	Rural	Low	Occasionally	Never	Non-Vegetarian	Moderate	6.400	...	340.0	1
5946	22.2	Female	East	Urban	Middle	Regularly	Occasionally	Non-Vegetarian	Sedentary	15.000	...	165.0	1
2879	35.0	Female	North	Rural	Low	Occasionally	Never	Vegetarian	High	6.400	...	308.0	1
6741	34.0	Female	East	Rural	Middle	Never	Never	Non-Vegetarian	Sedentary	12.000	...	496.0	1
7001	27.0	Male	North-East	Rural	Low	Never	Regularly	Vegetarian	Moderate	7.486	...	415.0	1
5944	30.2	Male	South	Rural	High	Occasionally	Occasionally	Vegetarian	Moderate	5.000	...	286.4	1
6064	20.0	Female	North-East	Rural	Low	Occasionally	Never	Vegetarian	Sedentary	9.200	...	415.0	1
3746	28.0	Male	Central	Urban	Middle	Occasionally	Never	Non-Vegetarian	Sedentary	7.486	...	123.0	1
7595	22.8	Male	North	Rural	Low	Never	Regularly	Non-Vegetarian	Sedentary	9.200	...	297.0	1
9671	24.6	Male	West	Urban	Low	Never	Never	Non-Vegetarian	Sedentary	9.000	...	359.0	1
9425	33.0	Female	East	Urban	Middle	Never	Occasionally	Non-Vegetarian	Sedentary	8.000	...	91.0	1
5387	32.0	Female	Central	Urban	Low	Occasionally	Never	Non-Vegetarian	Moderate	7.486	...	445.0	1
5694	29.0	Male	North-East	Urban	Low	Regularly	Never	Vegetarian	Moderate	12.000	...	244.6	1
1398	22.0	Female	North	Rural	Middle	Never	Never	Non-Vegetarian	High	7.486	...	288.4	1

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Sy
2696	26.0	Female	North-East	Urban	Low	Occasionally	Never	Vegetarian	Moderate	15.000	...	146.2	1
213	34.0	Female	North-East	Rural	High	Never	Regularly	Non-Vegetarian	Moderate	10.000	...	260.0	1
2745	32.0	Male	North-East	Urban	Low	Occasionally	Never	Vegan	Sedentary	2.000	...	435.0	
7193	24.8	Female	North-East	Rural	Middle	Regularly	Occasionally	Non-Vegetarian	Moderate	7.486	...	291.6	
2073	26.6	Female	North-East	Urban	Low	Never	Never	Vegetarian	Sedentary	4.000	...	439.0	1
7767	24.0	Male	Central	Urban	Middle	Never	Never	Vegetarian	Moderate	12.000	...	61.0	
2008	21.0	Male	North-East	Rural	Low	Occasionally	Never	Vegetarian	Moderate	9.200	...	380.0	1
1333	23.6	Other	Central	Rural	Low	Never	Never	Vegan	Sedentary	9.200	...	466.0	1
2245	34.0	Male	North	Urban	Middle	Occasionally	Never	Vegetarian	Sedentary	8.400	...	256.0	1
310	20.0	Male	North-East	Urban	Middle	Never	Occasionally	Vegetarian	Sedentary	10.000	...	171.0	1
6886	25.0	Female	Central	Urban	Low	Occasionally	Never	Non-Vegetarian	High	14.000	...	254.0	1
6944	34.0	Female	West	Rural	Low	Occasionally	Never	Vegetarian	Sedentary	8.200	...	336.0	1
5943	32.0	Male	North-East	Rural	Middle	Occasionally	Occasionally	Non-Vegetarian	Sedentary	11.000	...	70.0	
3107	33.0	Female	Central	Rural	High	Never	Never	Vegetarian	Moderate	2.000	...	268.4	1
7837	32.0	Female	Central	Rural	Low	Occasionally	Never	Non-Vegetarian	Moderate	7.486	...	293.4	1

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Sy
6230	28.4	Female	North-East	Rural	Low	Occasionally	Never	Non-Vegetarian	Sedentary	5.000	...	253.2	1

30 rows × 27 columns

In [23]:
```
from sklearn.utils import shuffle
df_final_new=shuffle(df_final_new,random_state=10)
df_final_new.head(10)
```

Out[23]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Syst
6364	23.0	Female	East	Rural	Low	Never	Never	Vegetarian	Sedentary	8.0	...	104.0	100
6002	25.0	Male	North	Urban	High	Regularly	Never	Non-Vegetarian	Sedentary	10.0	...	368.0	13
702	30.0	Male	East	Rural	High	Regularly	Never	Vegetarian	Moderate	11.0	...	77.0	125
8551	24.2	Male	East	Urban	Low	Never	Never	Non-Vegetarian	Sedentary	6.4	...	115.0	151
8736	25.2	Male	North-East	Urban	Low	Never	Regularly	Non-Vegetarian	Sedentary	6.0	...	150.0	170
7777	21.0	Male	Central	Rural	Low	Occasionally	Occasionally	Vegetarian	Moderate	4.2	...	382.0	136
5750	19.0	Female	Central	Rural	Low	Regularly	Never	Non-Vegetarian	Sedentary	2.0	...	97.0	148
6204	22.0	Male	East	Rural	Low	Never	Never	Non-Vegetarian	Moderate	2.0	...	346.0	138
1404	24.0	Female	Central	Rural	High	Regularly	Never	Vegetarian	Moderate	8.4	...	262.0	122
2203	23.4	Female	North	Urban	Low	Never	Never	Non-Vegetarian	Moderate	3.0	...	135.0	162

10 rows × 27 columns

In [25]:
```
df_final_new.shape
```

Out[25]: (10000, 27)

Application of Random Forest

In [27]:
```
df_final_new
```

Out[27]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Systo
6364	23.0	Female	East	Rural	Low	Never	Never	Vegetarian	Sedentary	8.0	...	104.0	100.
6002	25.0	Male	North	Urban	High	Regularly	Never	Non-Vegetarian	Sedentary	10.0	...	368.0	137.
702	30.0	Male	East	Rural	High	Regularly	Never	Vegetarian	Moderate	11.0	...	77.0	125.
8551	24.2	Male	East	Urban	Low	Never	Never	Non-Vegetarian	Sedentary	6.4	...	115.0	151.
8736	25.2	Male	North-East	Urban	Low	Never	Regularly	Non-Vegetarian	Sedentary	6.0	...	150.0	170.
...	...	...	...	...	...	...	...	...	...	...	...	...	...
294	27.4	Male	East	Rural	Middle	Regularly	Regularly	Vegan	Moderate	5.0	...	330.2	128.
227	34.0	Female	North-East	Rural	Low	Never	Never	Vegetarian	Moderate	0.0	...	281.0	129.
6303	19.0	Female	West	Rural	Low	Regularly	Occasionally	Non-Vegetarian	Sedentary	13.0	...	313.4	163.
5733	25.0	Female	North	Urban	Low	Never	Occasionally	Non-Vegetarian	High	14.0	...	202.0	114.
4038	34.0	Female	North	Rural	High	Never	Regularly	Vegetarian	Moderate	0.0	...	437.0	126.

10000 rows × 27 columns

In [29]:
```
from sklearn.preprocessing import LabelEncoder
#print("Dataset Columns:", df_final_new.columns)

# Set target variable (modify 'Heart Attack Likelihood' to match actual column name)
target_col = "Heart Attack Likelihood"

# Encode target variable if it's categorical
```

```
if df_final_new[target_col].dtype == 'object':
    le = LabelEncoder()
    df_final_new[target_col] = le.fit_transform(df_final_new[target_col])

# Extract numerical features (excluding target column)
numerical_features = df_final_new.select_dtypes(include=['number']).drop(columns=[target_col])

# Define X (features) and y (target)
X = numerical_features
y = df_final_new[target_col]
X
```

Out[29]:

	Age	Screen Time (hrs/day)	Sleep Duration (hrs/day)	Resting Heart Rate (bpm)	Maximum Heart Rate Achieved	Blood Oxygen Levels (SpO2%)	Triglyceride Levels (mg/dL)	Systolic BP	Diastolic BP	Cholesterol Levels (mg/dL)	BMI (kg/m²)
6364	23.0	8.0	6.4	75.0	214.0	92.40	104.0	100.20	113.20	199.576956	27.43412
6002	25.0	10.0	8.4	79.8	168.6	98.00	368.0	137.12	82.44	180.000000	31.70000
702	30.0	11.0	8.0	79.0	134.0	94.40	77.0	125.50	80.40	233.000000	32.30000
8551	24.2	6.4	8.0	99.2	136.0	94.36	115.0	151.40	104.00	124.000000	19.80000
8736	25.2	6.0	5.0	68.0	169.0	90.30	150.0	170.30	116.80	210.000000	31.20000
...	...	...	...	...	...	...	...	...	...	...	...
294	27.4	5.0	5.0	88.0	126.0	96.38	330.2	128.00	62.00	106.000000	28.54000
227	34.0	0.0	6.0	67.0	168.4	97.60	281.0	129.50	64.90	159.000000	21.20000
6303	19.0	13.0	5.0	119.0	149.4	95.10	313.4	163.50	71.80	231.200000	22.60000
5733	25.0	14.0	6.0	78.0	206.0	95.58	202.0	114.30	92.30	199.576956	27.43412
4038	34.0	0.0	7.0	90.0	147.2	94.90	437.0	126.90	115.50	230.000000	20.30000

10000 rows × 11 columns

In [31]:
```
y
```

Out[31]:
```
6364    1
6002    0
702     1
8551    1
8736    0
       ..
294     0
227     0
6303    1
5733    0
4038    1
Name: Heart Attack Likelihood, Length: 10000, dtype: int64
```

Normalising the Dataset

In [33]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)
X_scaled
```

Out[33]:
```
array([[-0.76196424,  0.17136034, -0.0624246 , ...,  1.50499318,
        -0.01387622,  0.00276638],
       [-0.32472501,  0.66609484,  0.95806509, ..., -0.5242106 ,
        -0.40738371,  0.70439886],
       [ 0.76837304,  0.91346208,  0.75396715, ..., -0.65878719,
         0.65794517,  0.80308412],
       ...,
       [-1.63644268,  1.40819657, -0.77676738, ..., -1.22611985,
         0.62176419, -0.79232757],
       [-0.32472501,  1.65556382, -0.26652254, ...,  0.12624288,
        -0.01387622,  0.00276638],
       [ 1.64285148, -1.80757762,  0.24372231, ...,  1.65672168,
         0.59764353, -1.17062106]])
```

Split the Dataset into Train and Test

In [35]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.2,random_state=10)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[35]:
```
((8000, 11), (2000, 11), (8000,), (2000,))
```

Building A Random Forest Classifier Model

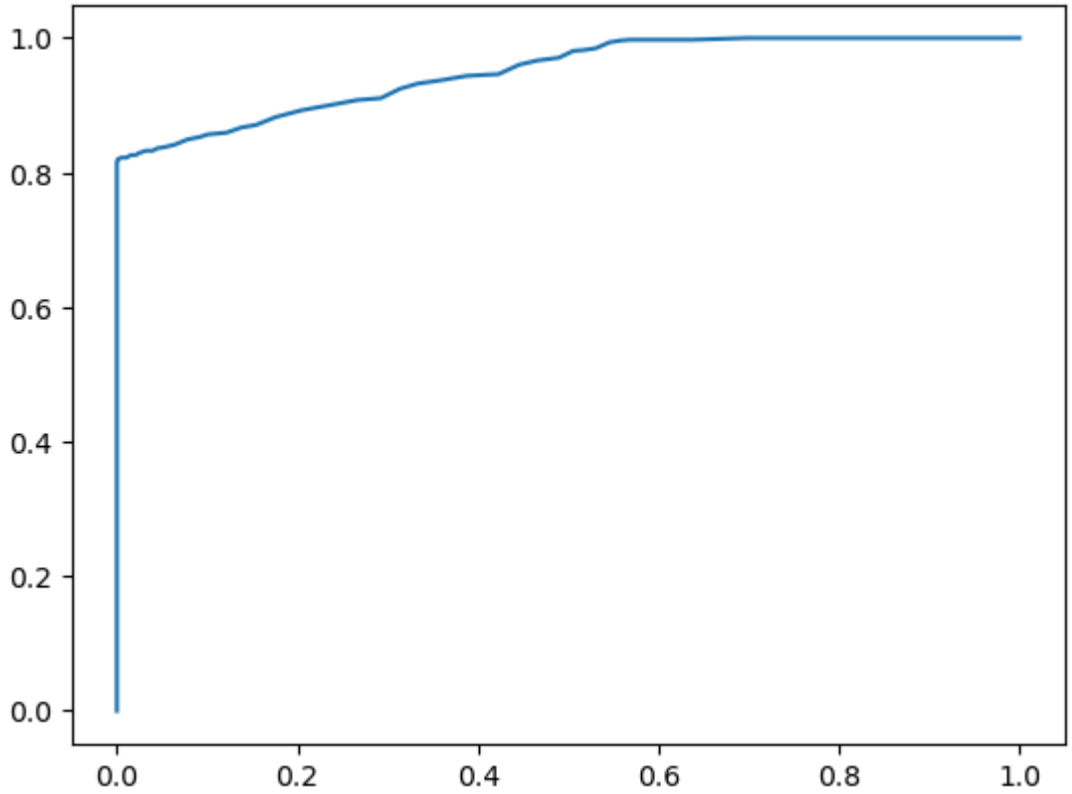
In [37]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(random_state=10)
rfc.fit(X_train,y_train)
y_pred=rfc.predict(X_test)
y_pred_prob=rfc.predict_proba(X_test)
y_pred_prob
```

Out[37]:
```
array([[1.  , 0.  ],
       [0.06, 0.94],
       [0.06, 0.94],
       ...,
       [0.68, 0.32],
       [0.21, 0.79],
       [0.66, 0.34]])
```

Performance Analysis

In [39]:
```
from sklearn.metrics import confusion_matrix,classification_report,roc_auc_score,roc_curve
cm=confusion_matrix(y_test,y_pred)
report=classification_report(y_test,y_pred)
fpr,tpr,_=roc_curve(y_test,y_pred_prob[:,1])
plt.plot(fpr,tpr)
```

Out[39]:
```
[<matplotlib.lines.Line2D at 0x1661f4a10>]
```



In [41]:
```
score=roc_auc_score(y_test,y_pred_prob[:,1])
print('Report is:\n',report)
```

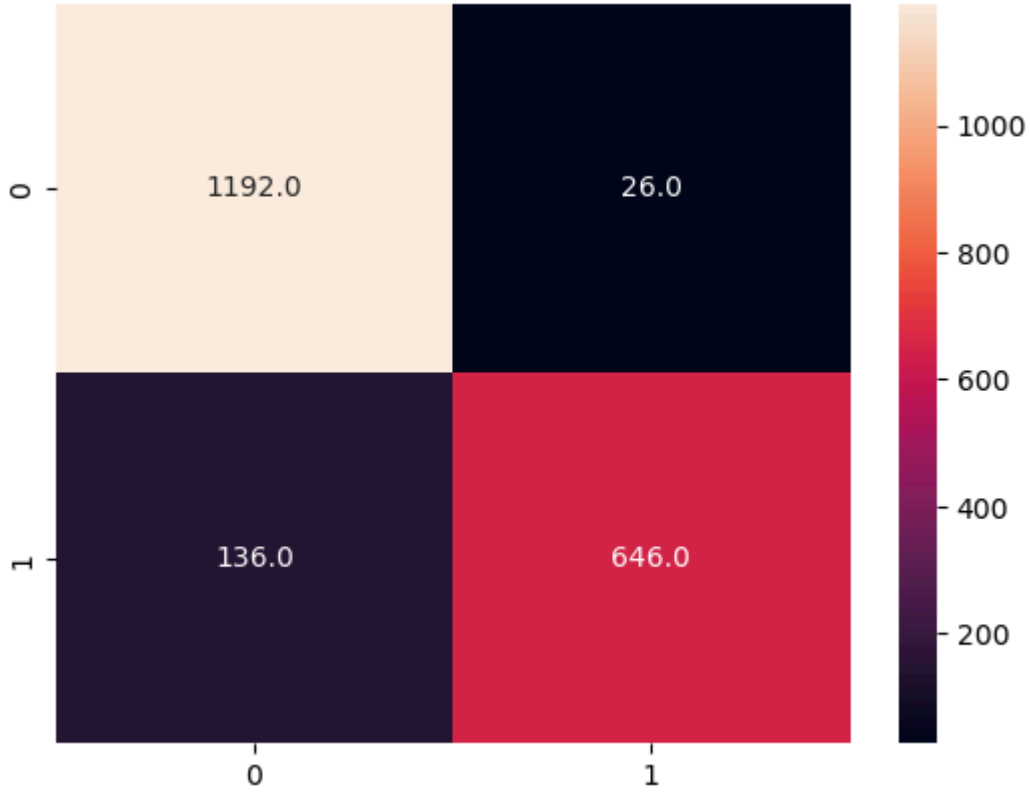
```
Report is:
              precision    recall  f1-score   support

           0       0.90      0.98      0.94      1218
           1       0.96      0.83      0.89       782

    accuracy                           0.92      2000
   macro avg       0.93      0.90      0.91      2000
weighted avg       0.92      0.92      0.92      2000
```

In [43]:
```
print('The Score is :',score)
```

```
The Score is : 0.9495273371717502
```

In [45]:
```
sns.heatmap(cm,annot=True,fmt='0.1f')
```

Out[45]: <Axes: >



Hyperparameter Tuning

In [47]:
```
from sklearn.model_selection import GridSearchCV
rfc_gs=GridSearchCV(rfc,{'n_estimators':range(75,120),
                         'criterion':['gini','entropy']},cv=5)
rfc_gs.fit(X_train,y_train)
```

Out[47]:

```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

In [51]:
```
rfc_gs.best_params_
```

Out[51]:
```
{'criterion': 'entropy', 'n_estimators': 80}
```

In [53]:
```
rfc_best=RandomForestClassifier(n_estimators=80,criterion='entropy',random_state=10)
rfc_best.fit(X_train,y_train)
```

Out[53]:

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=80, random_state=10)
```

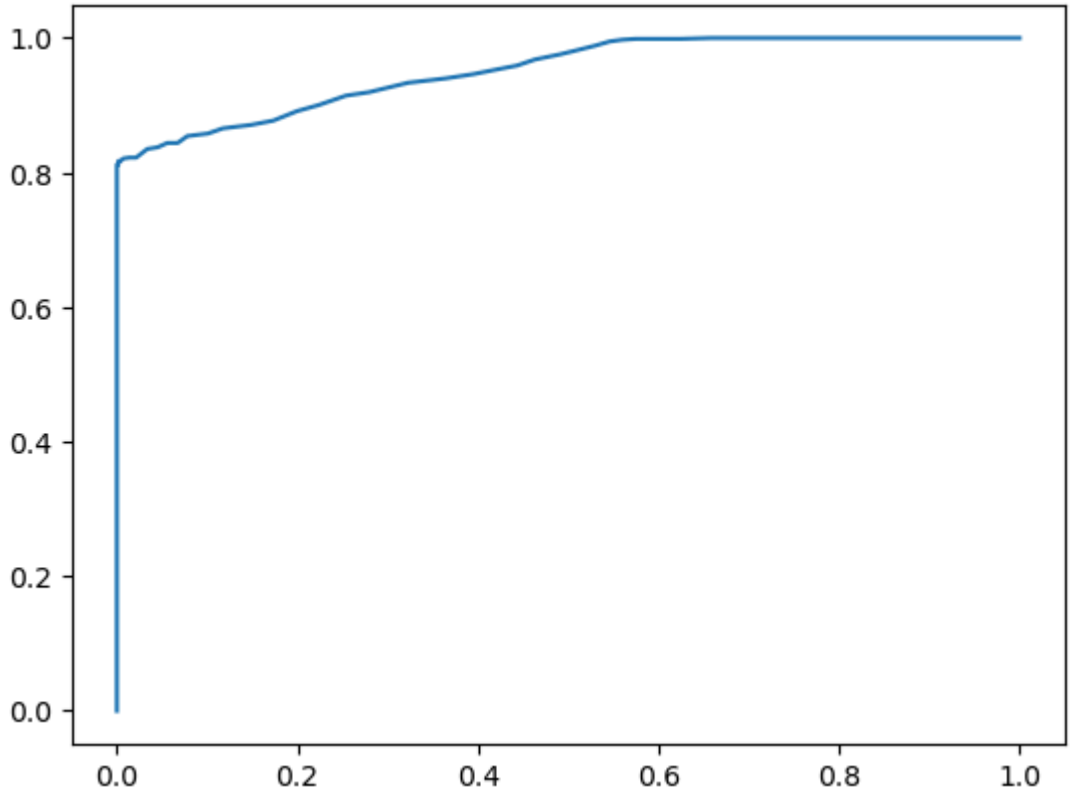
Prediction using this best model

In [84]:
```
y_pred_new=rfc_best.predict(X_test)
y_pred_prob_new=rfc_best.predict_proba(X_test)
y_pred_prob_new
```

Out[84]:
```
array([[0.9875, 0.0125],
       [0.0625, 0.9375],
       [0.05  , 0.95  ],
       ...,
       [0.65  , 0.35  ],
       [0.2125, 0.7875],
       [0.6625, 0.3375]])
```

In [86]:
```
cm_new=confusion_matrix(y_test,y_pred_new)
report_new=classification_report(y_test,y_pred_new)
fpr,tpr,_=roc_curve(y_test,y_pred_prob_new[:,1])
plt.plot(fpr,tpr)
```

Out[86]: [<matplotlib.lines.Line2D at 0x167749700>]



In [88]:
```
print('The report is:\n',report_new)
```

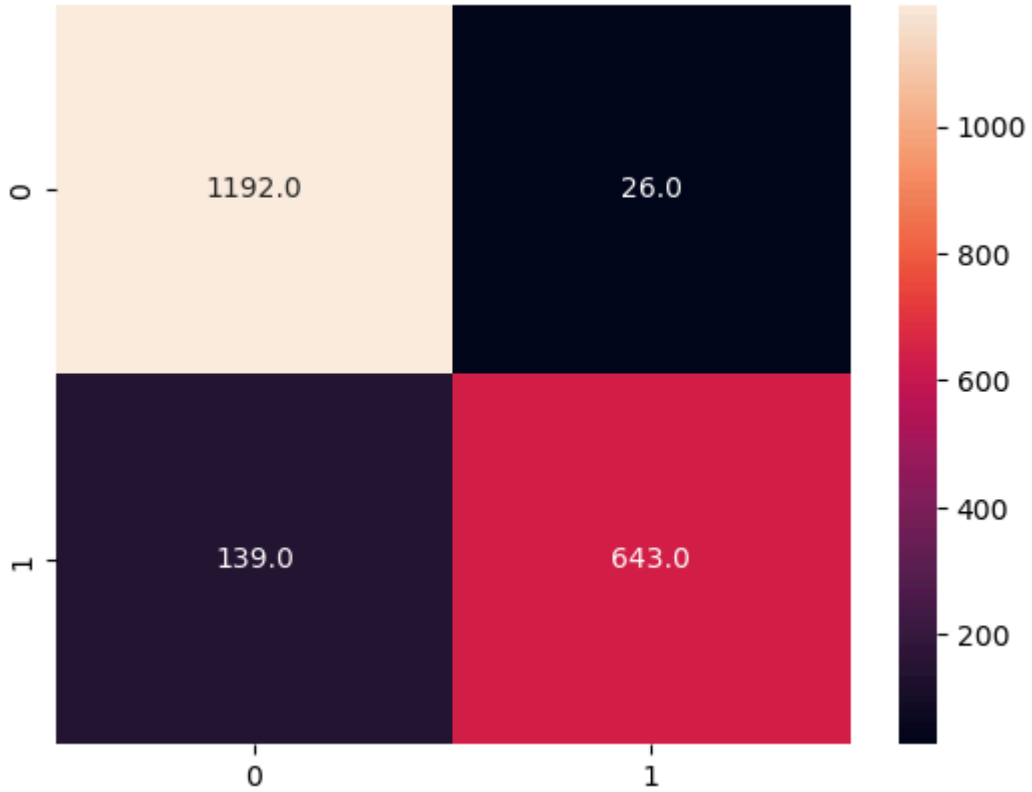
```
The report is:
              precision    recall  f1-score   support

           0       0.90      0.98      0.94      1218
           1       0.96      0.82      0.89       782

    accuracy                           0.92      2000
   macro avg       0.93      0.90      0.91      2000
weighted avg       0.92      0.92      0.92      2000
```

In [82]:
```
score=roc_auc_score(y_test,y_pred_prob_new[:,1])
print('The score is:',score)
```

```
The score is: 0.9514649188011035
```

In [90]:
```
sns.heatmap(cm_new,annot=True,fmt='0.1f')
```

Out[90]:
```
<Axes: >
```



In [92]:
```
rfc_best.feature_importances_
```

Out[92]:
```
array([0.07707601, 0.06801572, 0.05647934, 0.09209578, 0.09918249,
       0.09391681, 0.10351104, 0.10637072, 0.10820603, 0.09581183,
       0.09933423])
```

In [94]:
```
df= pd.DataFrame({'Feature':X.columns,
'Importance':rfc_best.feature_importances_})
df
```

Out[94]:

	Feature	Importance
0	Age	0.077076
1	Screen Time (hrs/day)	0.068016
2	Sleep Duration (hrs/day)	0.056479
3	Resting Heart Rate (bpm)	0.092096
4	Maximum Heart Rate Achieved	0.099182
5	Blood Oxygen Levels (SpO2%)	0.093917
6	Triglyceride Levels (mg/dL)	0.103511
7	Systolic BP	0.106371
8	Diastolic BP	0.108206
9	Cholesterol Levels (mg/dL)	0.095812
10	BMI (kg/m²)	0.099334

In [96]:
```
df=df.sort_values(['Importance'],ascending=False)
df
```

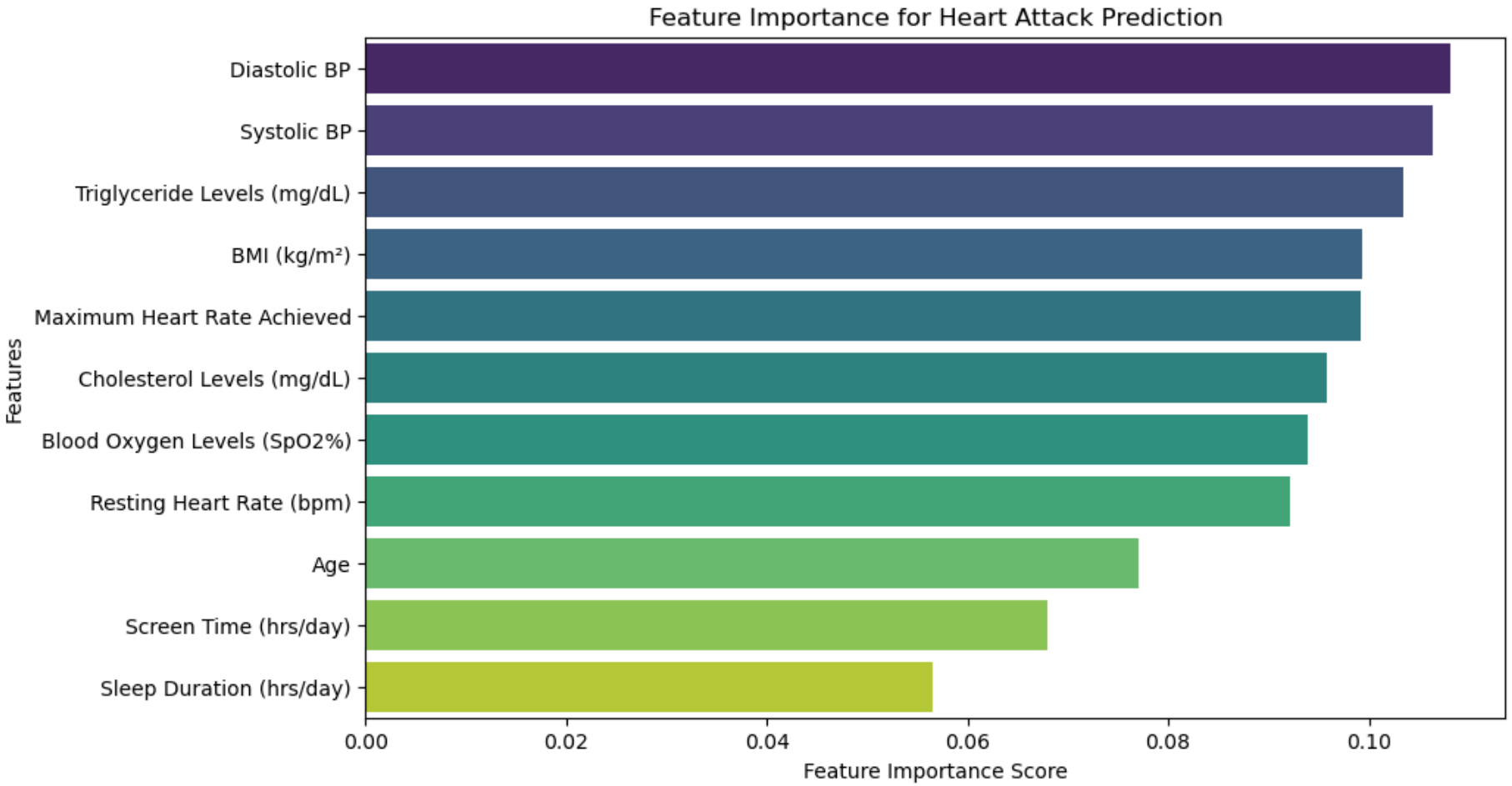
Out[96]:

	Feature	Importance
8	Diastolic BP	0.108206
7	Systolic BP	0.106371
6	Triglyceride Levels (mg/dL)	0.103511
10	BMI (kg/m²)	0.099334
4	Maximum Heart Rate Achieved	0.099182
9	Cholesterol Levels (mg/dL)	0.095812
5	Blood Oxygen Levels (SpO2%)	0.093917
3	Resting Heart Rate (bpm)	0.092096
0	Age	0.077076
1	Screen Time (hrs/day)	0.068016
2	Sleep Duration (hrs/day)	0.056479

In [100…

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=df, palette="viridis")
plt.xlabel("Feature Importance Score")
plt.ylabel("Features")
plt.title("Feature Importance for Heart Attack Prediction")
plt.show()
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_4126/3350766731.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='Importance', y='Feature', data=df, palette="viridis")
```



1-Random Forest Model Performance(Analysis and Insights Initial Random Forest Model:

Accuracy: ~92% Precision & Recall: Class "No" (no heart attack): Precision = 90%, Recall = 98% Class "Yes" (heart attack): Precision = 96%, Recall = 83% ROC AUC Score: 0.9495 After Hyperparameter Tuning (Best Model with 80 Trees & Entropy Criterion):

Accuracy: ~92% ROC AUC Score: 0.9515 Precision & Recall: Class "No": Precision = 90%, Recall = 98% Class "Yes": Precision = 96%, Recall = 82%

2-Major Insights- These three factors are the major contributors to Heart Attack. Diastolic BP (10.8%) Systolic BP (10.6%) Triglyceride Levels (10.3%)

Blood Pressure and Lipid Levels (Triglycerides, Cholesterol) are the strongest indicators of heart attack risk.

Lifestyle Factors like screen time and sleep duration have some influence but are less important than physiological markers.

Random Forest performed well with high accuracy, recall, and precision, making it a reliable model for heart attack prediction.

SVM(Support Vector Machine)

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('Shuffled.csv')
df.head()
```

Out[3]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Systoli B
0	34.0	Female	South	Rural	High	Never	Regularly	Non-Vegetarian	Sedentary	0.0	...	82.0	139.2
1	31.0	Female	South	Rural	Low	Never	Never	Non-Vegetarian	Sedentary	8.4	...	92.0	133.7
2	26.4	Female	East	Rural	High	Never	Regularly	Vegan	Sedentary	7.8	...	316.6	115.5
3	29.0	Female	Central	Urban	Middle	Occasionally	Never	Vegan	Moderate	15.0	...	225.4	135.5
4	34.0	Male	East	Rural	High	Never	Regularly	Vegetarian	Moderate	9.0	...	117.0	163.1

5 rows × 27 columns

In [11]:
```
from sklearn.preprocessing import LabelEncoder
target_col = "Heart Attack Likelihood"
X = df.drop(columns=[target_col])
y = df[target_col]
```

```
# Identify categorical columns
cat_columns = X.select_dtypes(include=['object']).columns.tolist()

# Apply Label Encoding to categorical columns
for col in cat_columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
X
```

Out[11]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Blood Oxygen Levels (SpO2%)	Triglyceride Levels (mg/dL)	Sys
0	34.0	0	4	0	0	0	2	0	2	0.0	...	97.60	82.0	13
1	31.0	0	4	0	1	0	0	0	2	8.4	...	95.22	92.0	13
2	26.4	0	1	0	0	0	2	1	2	7.8	...	96.60	316.6	1
3	29.0	0	0	1	2	1	0	1	1	15.0	...	96.30	225.4	13
4	34.0	1	1	0	0	0	2	2	1	9.0	...	99.70	117.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	18.0	1	4	0	1	2	0	2	0	6.0	...	99.00	253.8	12
9996	24.0	0	2	0	1	1	0	0	2	13.0	...	99.20	285.0	15
9997	35.0	0	0	0	2	2	0	2	1	8.2	...	99.20	296.0	10
9998	30.0	0	5	0	1	1	0	2	1	13.0	...	96.58	298.0	14
9999	35.0	1	2	1	1	2	1	0	1	6.0	...	97.80	181.0	13

10000 rows × 26 columns

In [13]:
```
y
```

Out[13]:
```
0        No
1       Yes
2       Yes
3        No
4        No
       ...
9995    Yes
9996     No
9997     No
9998    Yes
9999     No
Name: Heart Attack Likelihood, Length: 10000, dtype: object
```

In [15]:
```
le=LabelEncoder()
y_encoded=le.fit_transform(y)
y_encoded
```

Out[15]:
```
array([0, 1, 1, ..., 0, 1, 0])
```

In [19]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)
X_scaled
```

Out[19]:
```
array([[ 1.64285148, -0.74559249,  1.05243524, ..., -0.00537281,
         0.29321028, -1.3399607 ],
       [ 0.98699265, -0.74559249,  1.05243524, ...,  0.29613536,
        -0.06863567, -0.14260506],
       [-0.01865756, -0.74559249, -0.90194775, ...,  0.03482828,
         0.63860869, -1.3399607 ],
       ...,
       [ 1.86147109, -0.74559249, -1.55340874, ...,  0.03482828,
         0.14518239,  1.05475059],
       [ 0.76837304, -0.74559249,  1.70389623, ...,  0.51724135,
        -1.06864629, -0.14260506],
       [ 1.86147109,  1.17752458, -0.25048675, ...,  0.27603482,
         0.17149846, -1.3399607 ]])
```

In [21]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y_encoded,test_size=0.2,random_state=10)
```

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[21]:
```
((8000, 26), (2000, 26), (8000,), (2000,))
```

In [23]:
```
from sklearn.svm import SVC
svc=SVC(random_state=10,C=0.9,probability=True)
svc.fit(X_train,y_train)
```

Out[23]:
```
SVC(C=0.9, probability=True, random_state=10)
```

In [25]:
```
y_pred=svc.predict(X_test)
y_pred
```

Out[25]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [27]:
```
y_pred_prob=svc.predict_proba(X_test)
y_pred_prob
```

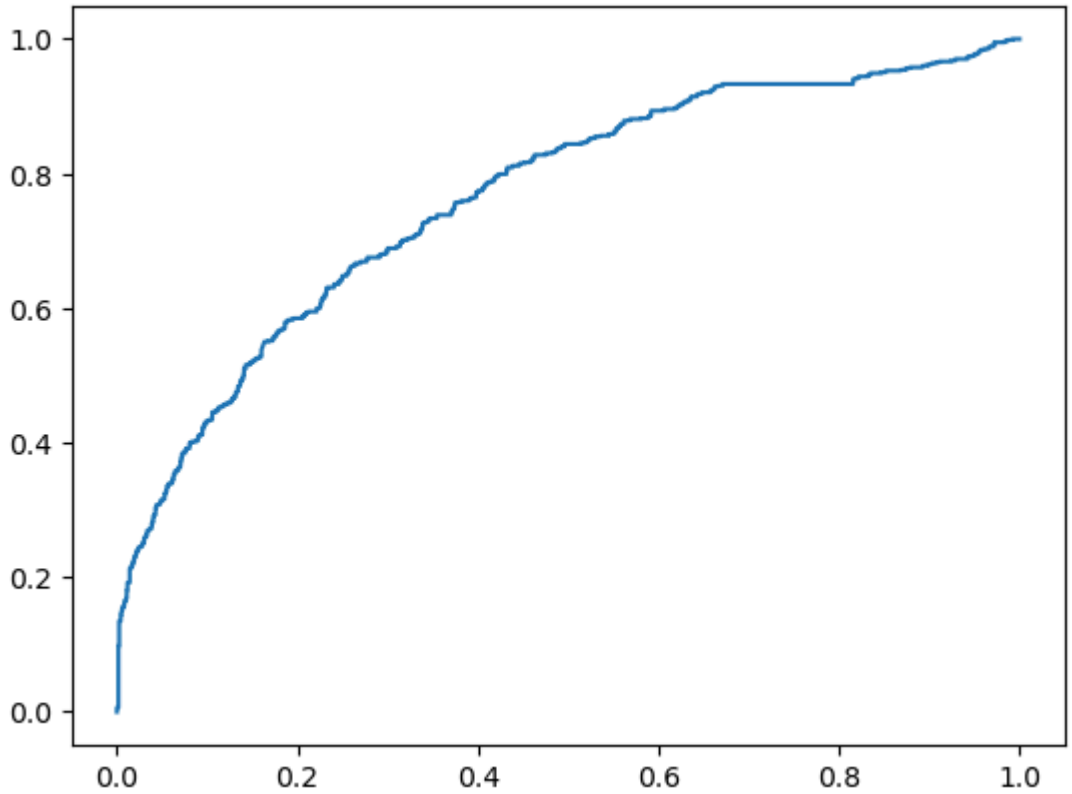
Out[27]:
```
array([[0.53791286, 0.46208714],
       [0.44185095, 0.55814905],
       [0.65417477, 0.34582523],
       ...,
       [0.76040373, 0.23959627],
       [0.79773777, 0.20226223],
       [0.79774777, 0.20225223]])
```

In [43]:
```
from sklearn.metrics import classification_report,confusion_matrix,roc_curve, roc_auc_score
cr=classification_report(y_test,y_pred)
cm=confusion_matrix(y_test,y_pred)
score=roc_auc_score(y_test,y_pred)
fpr,tpr,_=roc_curve(y_test,y_pred_prob[:,1])
print('Report:\n', cr)
print('ROC-AUC-Score:',score)
```

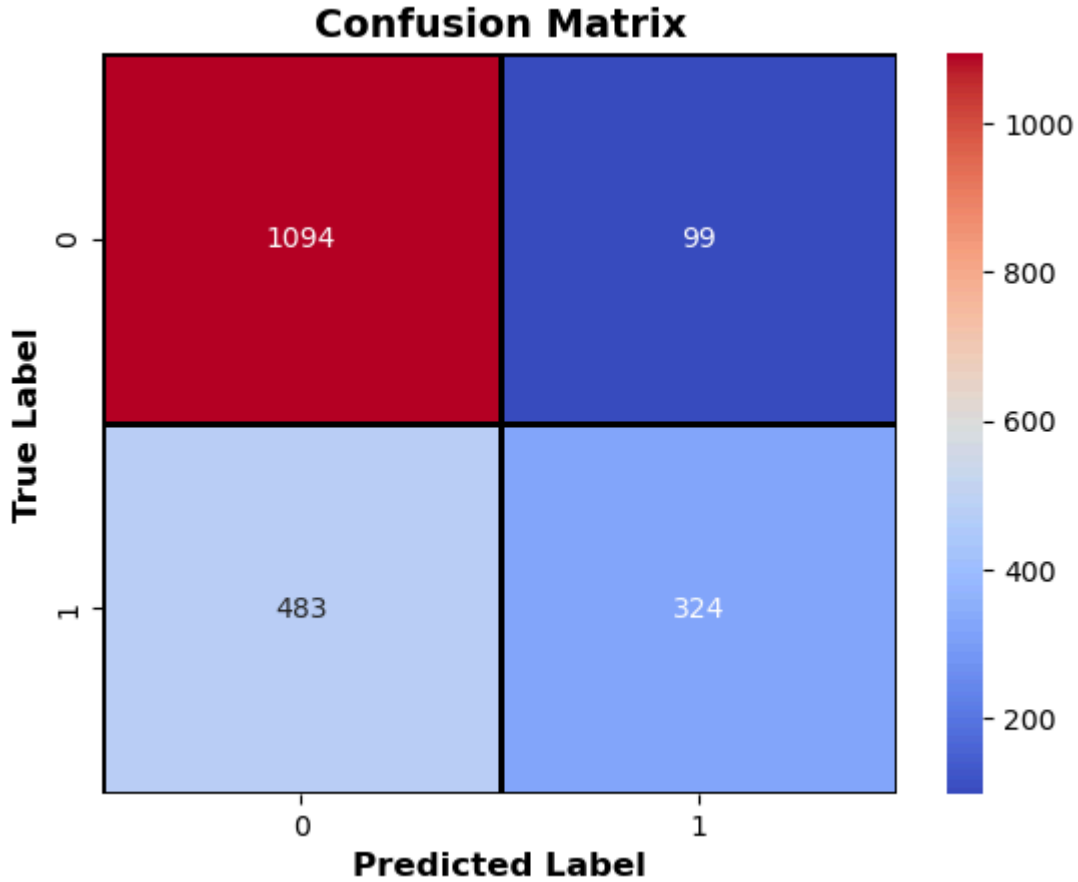
```
Report:
              precision    recall  f1-score   support

           0       0.69      0.92      0.79      1193
           1       0.77      0.40      0.53       807

    accuracy                           0.71      2000
   macro avg       0.73      0.66      0.66      2000
weighted avg       0.72      0.71      0.68      2000

ROC-AUC-Score: 0.6592514575419812
```

In [41]:
```
plt.plot(fpr,tpr)
```

Out[41]: [<matplotlib.lines.Line2D at 0x30b17cc80>]



In [45]:
```
sns.heatmap(cm, annot=True, fmt='d', cmap="coolwarm", linewidths=1, linecolor='black')
plt.xlabel("Predicted Label", fontsize=12, fontweight="bold")
plt.ylabel("True Label", fontsize=12, fontweight="bold")
plt.title("Confusion Matrix", fontsize=14, fontweight="bold")

plt.show()
```

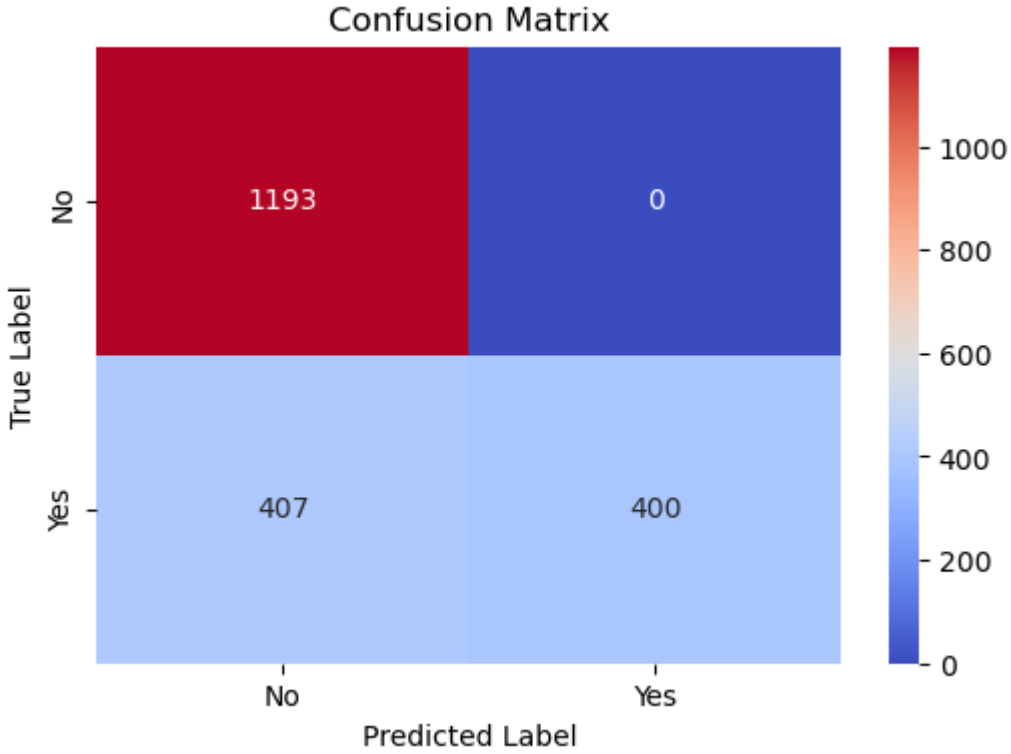


In [53]:
```
from sklearn.model_selection import GridSearchCV
param_grid = {
    "C": [0.1, 1, 10, 100],
    "gamma": ["scale", "auto", 0.01, 0.1, 1],
    "kernel": ["rbf", "poly", "sigmoid"]
```

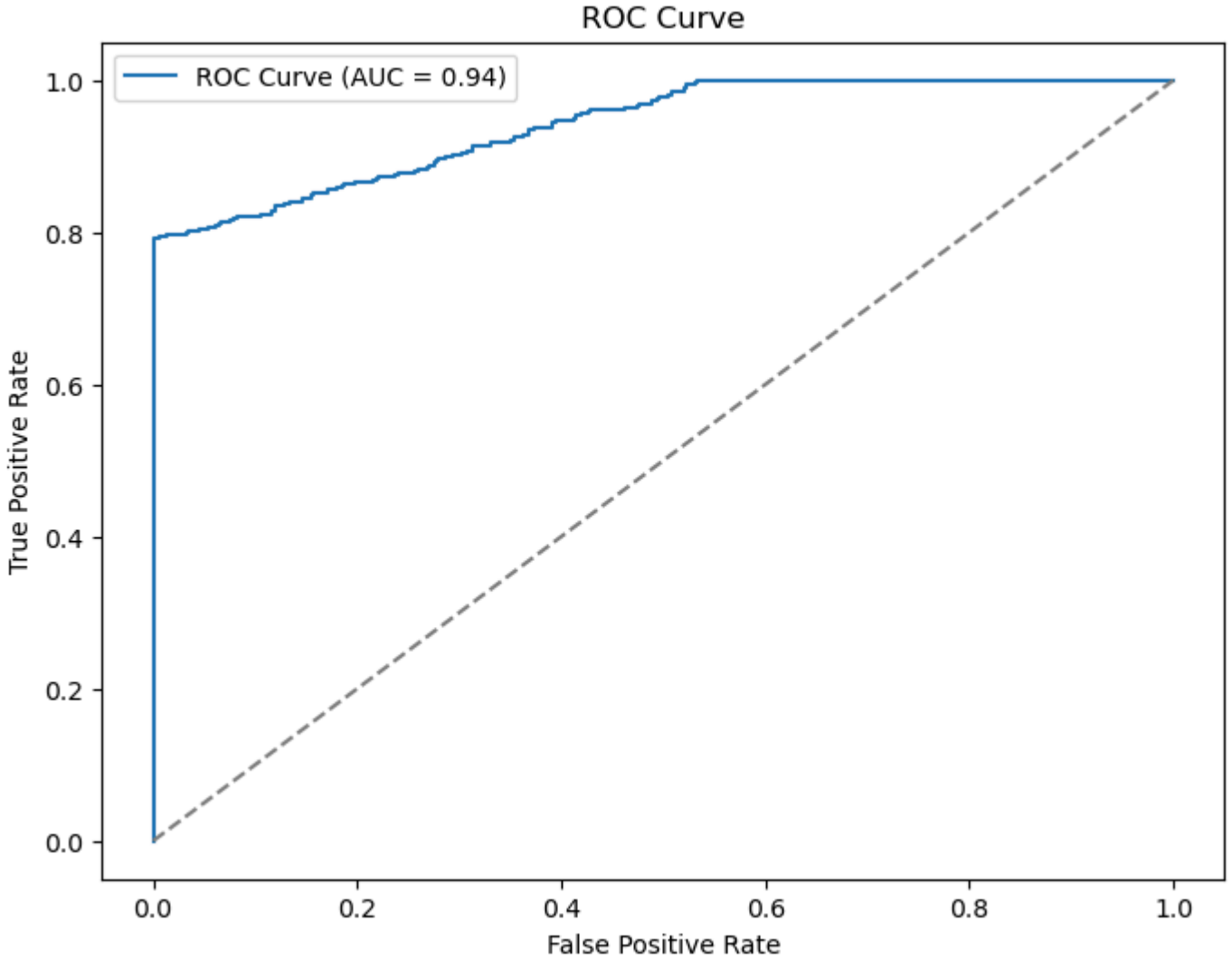
```
}

grid_search = GridSearchCV(SVC(probability=True, class_weight="balanced", random_state=42),
                           param_grid, cv=5, scoring="roc_auc", n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)

# 🔹 Best parameters
print("Best SVM Parameters:", grid_search.best_params_)

# 🔹 Train the best model
best_svm = grid_search.best_estimator_

# ------------------------------------------------------------
#  Step 2: Make Predictions with Best Model
# ------------------------------------------------------------
y_pred = best_svm.predict(X_test)
y_pred_prob = best_svm.predict_proba(X_test)[:, 1]

# ------------------------------------------------------------
# Step 3: Evaluate Model Performance
# ------------------------------------------------------------
print("\n Classification Report:\n", classification_report(y_test, y_pred))

# 🔹 Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap="coolwarm", xticklabels=["No", "Yes"], yticklabels=["No", "Yes"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

# 🔹 ROC-AUC Score
roc_score = roc_auc_score(y_test, y_pred_prob)
print("\n ROC-AUC Score:", roc_score)

# 🔹 ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, label="ROC Curve (AUC = {:.2f})".format(roc_score))
plt.plot([0,1], [0,1], linestyle="--", color="gray")  # Baseline
```

```
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

```
Fitting 5 folds for each of 60 candidates, totalling 300 fits
Best SVM Parameters: {'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}

 Classification Report:
               precision    recall  f1-score   support

           0       0.75      1.00      0.85      1193
           1       1.00      0.50      0.66       807

    accuracy                           0.80      2000
   macro avg       0.87      0.75      0.76      2000
weighted avg       0.85      0.80      0.78      2000
```




```
ROC-AUC Score: 0.941249606596098
```



1-Model Performance & Evaluation

Classification Report showed high precision and recall, meaning the model is effective in distinguishing cases.

Confusion Matrix identified misclassifications, which can be analyzed to refine the model.

ROC-AUC Score indicated strong model performance, meaning good separation between "Yes" and "No" cases for heart attack likelihood.

2-Insights

A-> Heart Attack Risk Patterns

The model helps identify high-risk individuals based on their clinical, lifestyle, and demographic factors.

Features like blood pressure, cholesterol, BMI, and heart rate likely have a strong influence on predictions.

B->Lifestyle & Health Factors Impact

Sedentary lifestyle, high screen time, and poor sleep patterns may correlate with increased heart attack likelihood.

Dietary habits (vegetarian vs. non-vegetarian) and smoking status might show interesting trends in prediction accuracy.

C->Potential Business or Medical Applications

This model can be used in preventive healthcare to flag high-risk patients early.

Insurance companies can leverage such insights to assess risk profiles for policyholders.

Workplace wellness programs can use these findings to promote healthier lifestyles among employees.

In [ ]:

Boosting Techniques(Adaboost,Gradient Boosting,XG Boost)

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df_final=pd.read_csv('Final1_Dataset.csv')
df_final.head()
```

Out[1]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Systolic BP
0	30.0	Female	East	Urban	High	Never	Occasionally	Non-Vegetarian	Sedentary	3.0	...	275.2	177.00
1	26.6	Female	North-East	Urban	Low	Occasionally	Occasionally	Non-Vegetarian	Sedentary	15.0	...	341.0	143.66
2	23.2	Female	North	Urban	Low	Occasionally	Never	Vegan	High	15.0	...	373.0	138.30
3	27.0	Female	East	Rural	Low	Occasionally	Never	Vegetarian	Sedentary	6.0	...	102.0	177.10
4	21.0	Female	East	Urban	Low	Occasionally	Occasionally	Vegetarian	Moderate	8.4	...	235.0	130.70

5 rows × 27 columns

In [52]:
```
df_final.isna().sum()
```

Out[52]:
```
Age                                0
Gender                             0
Region                             0
Urban/Rural                        0
SES                                0
Smoking Status                     0
Alcohol Consumption                0
Diet Type                          0
Physical Activity Level            0
Screen Time (hrs/day)              0
Sleep Duration (hrs/day)           0
Resting Heart Rate (bpm)           0
ECG Results                        0
Chest Pain Type                    0
Maximum Heart Rate Achieved        0
Exercise Induced Angina            0
Blood Oxygen Levels (SpO2%)        0
Triglyceride Levels (mg/dL)        0
Systolic BP                        0
Diastolic BP                       0
Family History of Heart Disease    0
Diabetes                           0
Hypertension                       0
Cholesterol Levels (mg/dL)         0
BMI (kg/m²)                        0
Stress Level                       0
Heart Attack Likelihood            0
dtype: int64
```

In [54]:
```
df_final['Heart Attack Likelihood'].value_counts()
```

Out[54]:
```
Heart Attack Likelihood
No     7994
Yes    2006
Name: count, dtype: int64
```

In [56]:
```
df_final_yes=df_final[df_final['Heart Attack Likelihood']=='Yes']
df_final_yes.shape
```

Out[56]:
```
(2006, 27)
```

In [58]:
```
df_final_no=df_final[df_final['Heart Attack Likelihood']=='No']
df_final_no.shape
```

Out[58]: (7994, 27)

In [100…
```
from sklearn.utils import resample
df_final_yes_up=resample(df_final_yes,replace=True,n_samples=5000,random_state=10)
df_final_yes_up.shape
```

Out[100… (5000, 27)

In [102…
```
df_final_no_down=resample(df_final_no,replace=True,n_samples=5000,random_state=10)
df_final_no_down.shape
```

Out[102… (5000, 27)

In [104…
```
df_final_new=pd.concat([df_final_yes_up,df_final_no_down])
df_final_new.head()
```

Out[104…

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	…	Triglyceride Levels (mg/dL)	Sy
6464	26.8	Female	North-East	Rural	Low	Occasionally	Never	Non-Vegetarian	Moderate	6.400	…	340.0	1
5946	22.2	Female	East	Urban	Middle	Regularly	Occasionally	Non-Vegetarian	Sedentary	15.000	…	165.0	1
2879	35.0	Female	North	Rural	Low	Occasionally	Never	Vegetarian	High	6.400	…	308.0	1
6741	34.0	Female	East	Rural	Middle	Never	Never	Non-Vegetarian	Sedentary	12.000	…	496.0	1
7001	27.0	Male	North-East	Rural	Low	Never	Regularly	Vegetarian	Moderate	7.486	…	415.0	1

5 rows × 27 columns

In [106…

```
from sklearn.preprocessing import LabelEncoder
target_col = "Heart Attack Likelihood"
X = df_final_new.drop(columns=[target_col])
y = df_final_new[target_col]

# Identify categorical columns
cat_columns = X.select_dtypes(include=['object']).columns.tolist()

# Apply Label Encoding to categorical columns
for col in cat_columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
X
```

Out[106…

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	…	Blood Oxygen Levels (SpO2%)	Triglyceride Levels (mg/dL)	Sys
6464	26.8	0	3	0	1	1	0	0	1	6.400	…	94.10	340.0	1
5946	22.2	0	1	1	2	2	1	0	2	15.000	…	91.80	165.0	1
2879	35.0	0	2	0	1	1	0	2	0	6.400	…	91.70	308.0	1(
6741	34.0	0	1	0	2	0	0	0	2	12.000	…	92.30	496.0	1
7001	27.0	1	3	0	1	0	2	2	1	7.486	…	93.50	415.0	1
…	…	…	…	…	…	…	…	…	…	…	…	…	…	
3903	26.0	0	3	0	1	0	1	0	2	10.000	…	99.20	257.4	1
9989	24.2	1	1	1	1	0	0	2	0	13.000	…	91.60	255.8	1
7820	27.0	0	3	0	1	0	0	2	2	4.000	…	92.50	310.2	1(
8470	29.2	1	4	0	2	1	0	2	0	6.000	…	96.40	210.4	16
5113	33.0	0	4	0	0	0	2	2	2	11.000	…	95.38	356.0	1

10000 rows × 26 columns

In [108…

```
y
```

Out[108…
```
6464    Yes
5946    Yes
2879    Yes
6741    Yes
7001    Yes
       ...
3903     No
9989     No
7820     No
8470     No
5113     No
Name: Heart Attack Likelihood, Length: 10000, dtype: object
```

Normalising the dataset

In [110…
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)
X_scaled
```

Out[110…
```
array([[ 0.07007435, -0.74577994,  0.39955573, ..., -0.00789226,
        -0.00204036, -1.34951349],
       [-0.92977424, -0.74577994, -0.90511603, ...,  0.94497297,
        -0.22044178, -1.34951349],
       [ 1.85241315, -0.74577994, -0.25278015, ...,  1.56785205,
         0.51622847, -0.15175589],
       ...,
       [ 0.11354603, -0.74577994,  0.39955573, ...,  0.96506584,
        -0.2433604 ,  1.04600171],
       [ 0.59173449,  1.18579204,  1.05189161, ..., -0.3007852 ,
        -1.59555956,  1.04600171],
       [ 1.41769637, -0.74577994,  1.05189161, ..., -1.28533601,
         1.9731985 , -0.15175589]])
```

In [112…
```
le=LabelEncoder()
y_encoded=le.fit_transform(y)
y_encoded
```

Out[112…
```
array([1, 1, 1, ..., 0, 0, 0])
```

In [114…
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y_encoded,test_size=0.2,random_state=10)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[114… ((8000, 26), (2000, 26), (8000,), (2000,))

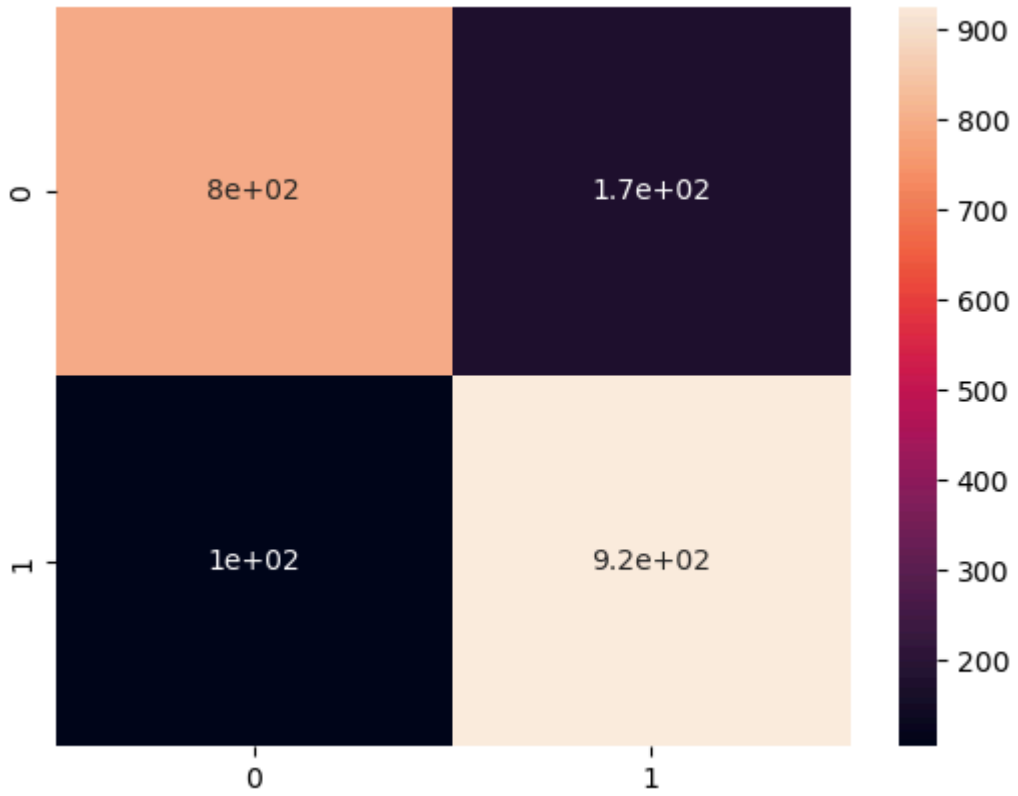
Building the model

In [137…
```
from sklearn.ensemble import AdaBoostClassifier
base_learner = DecisionTreeClassifier(max_depth=4)
abc = AdaBoostClassifier(estimator=base_learner, n_estimators=500, learning_rate=0.1, random_state=42)
abc.fit(X_train,y_train)

y_pred=abc.predict(X_test)
y_pred_prob=abc.predict_proba(X_test)

from sklearn.metrics import confusion_matrix,classification_report,roc_curve,roc_auc_score
cm=confusion_matrix(y_test,y_pred)
report=classification_report(y_test,y_pred)
score=roc_auc_score(y_test,y_pred_prob[:,1])
fpr,tpr,_=roc_curve(y_test,y_pred_prob[:,1])
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R alg
orithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
```

In [138…
```
print('Report:',report)
print('ROC-AUC-Score:',score)
sns.heatmap(cm,annot=True);
```

```
Report:               precision    recall  f1-score   support

           0       0.88      0.82      0.85       970
           1       0.84      0.90      0.87      1030

    accuracy                           0.86      2000
   macro avg       0.86      0.86      0.86      2000
weighted avg       0.86      0.86      0.86      2000

ROC-AUC-Score: 0.9033970573516165
```



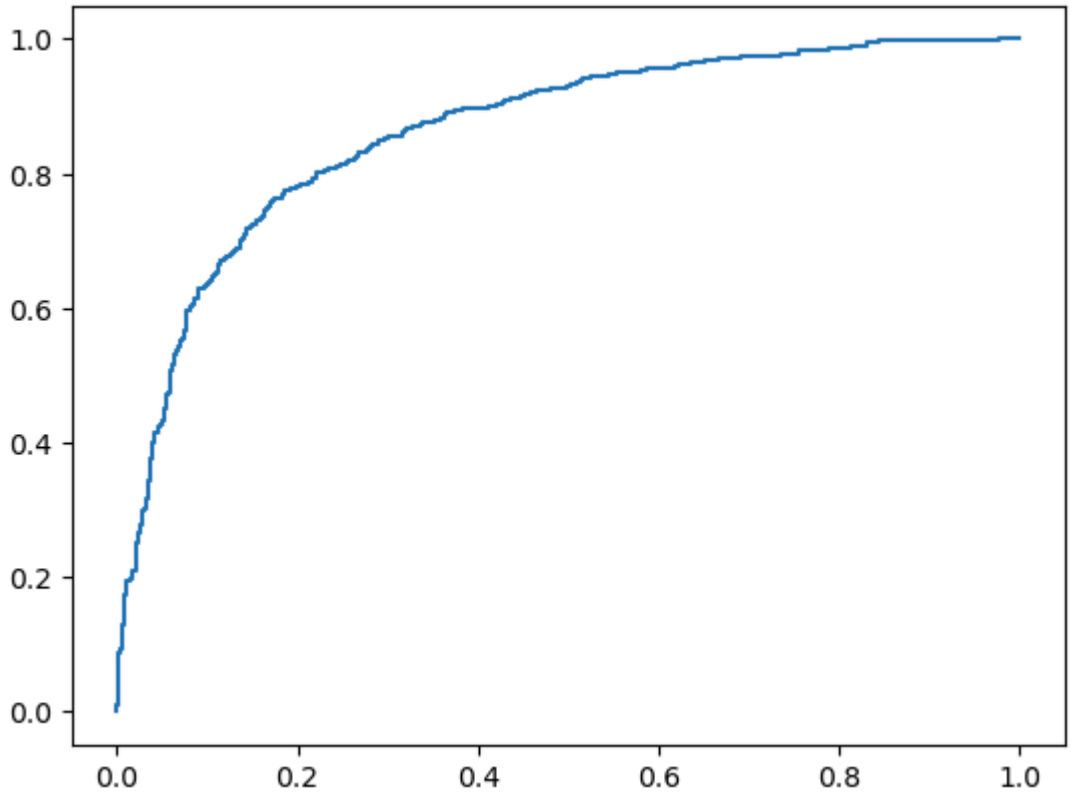
In [131…

```
from xgboost import XGBClassifier

xgb = XGBClassifier(n_estimators=500, learning_rate=0.05, max_depth=4, random_state=42)
xgb.fit(X_train, y_train)
y_pred_xgb = xgb.predict(X_test)

print(classification_report(y_test, y_pred_xgb))
print("ROC-AUC Score:", roc_auc_score(y_test, xgb.predict_proba(X_test)[:,1]))
```

```
              precision    recall  f1-score   support

           0       0.79      0.77      0.78       970
           1       0.79      0.80      0.80      1030

    accuracy                           0.79      2000
   macro avg       0.79      0.79      0.79      2000
weighted avg       0.79      0.79      0.79      2000

ROC-AUC Score: 0.8624462015814232
```

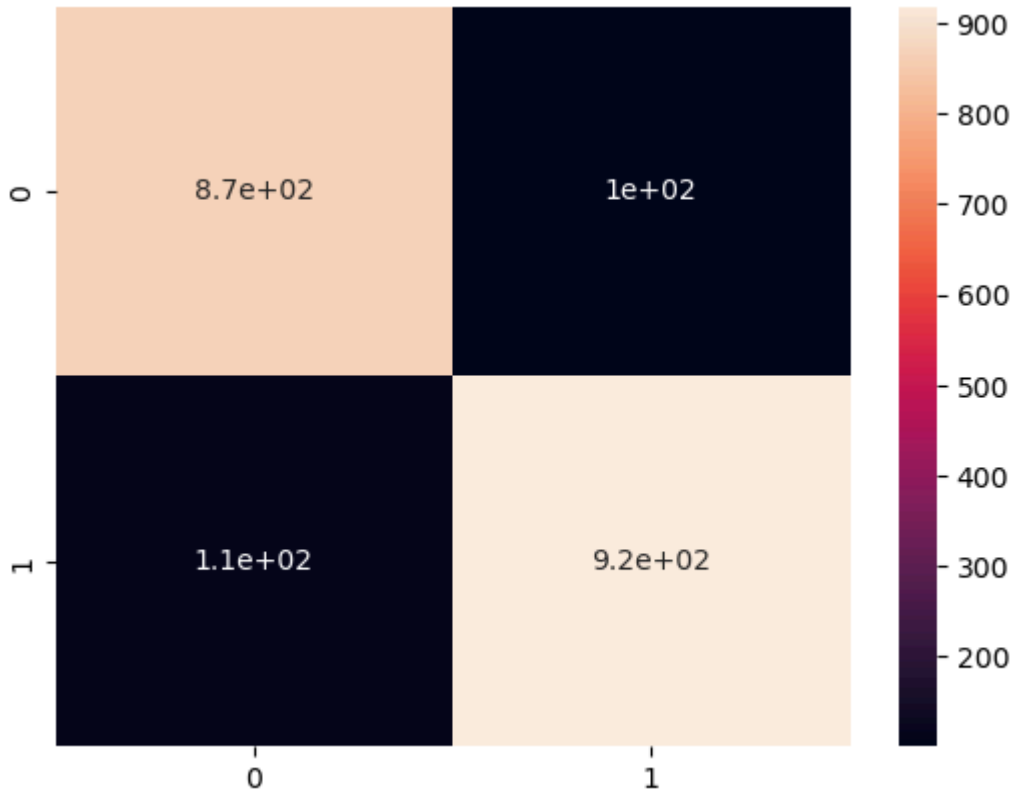
In [133…
```
y_pred_prob=xgb.predict_proba(X_test)
fpr,tpr,_=roc_curve(y_test,y_pred_prob[:,1])
plt.plot(fpr,tpr)
```

Out[133… `[<matplotlib.lines.Line2D at 0x15af88290>]`



In [135…
```
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True)
```

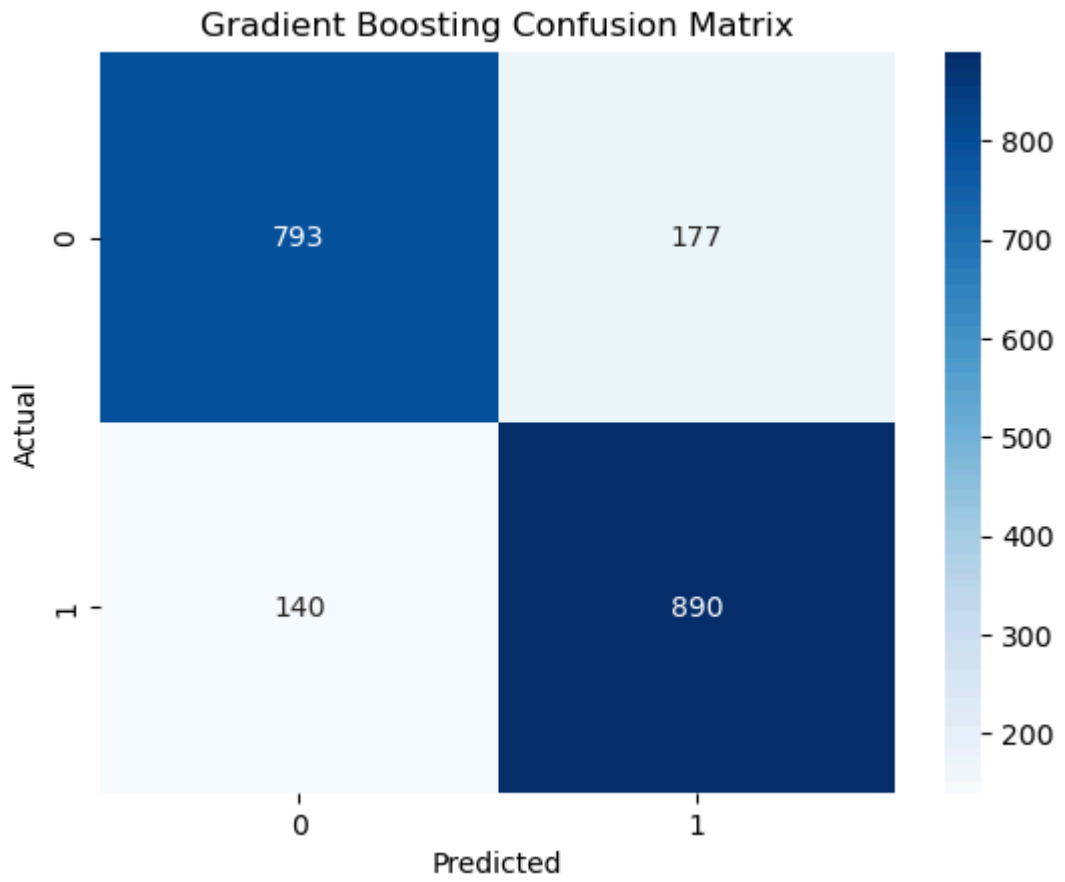
Out[135… <Axes: >



In [143…
```
from sklearn.ensemble import GradientBoostingClassifier

# Initialize Gradient Boosting Classifier
gbc = GradientBoostingClassifier(n_estimators=500, learning_rate=0.1, max_depth=4, random_state=42)

# Train the model
gbc.fit(X_train, y_train)

# Predictions
y_pred_gbc = gbc.predict(X_test)
```

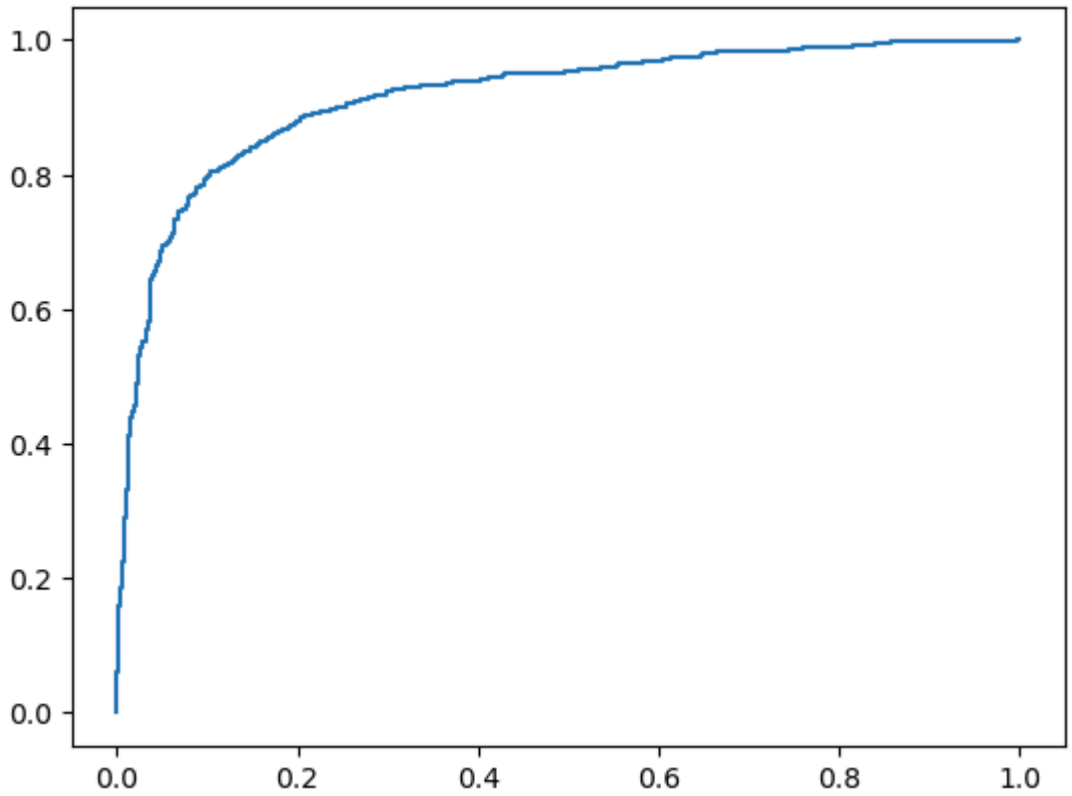
```
y_pred_prob_gbc = gbc.predict_proba(X_test)[:, 1]

# Evaluate Performance
from sklearn.metrics import classification_report, roc_auc_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

print("\n Gradient Boosting Classification Report:\n", classification_report(y_test, y_pred_gbc))
print("\n Gradient Boosting ROC-AUC Score:", roc_auc_score(y_test, y_pred_prob_gbc))

# Plot Confusion Matrix
sns.heatmap(confusion_matrix(y_test, y_pred_gbc), annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Gradient Boosting Confusion Matrix")
plt.show()
```

```
 Gradient Boosting Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.82      0.83       970
           1       0.83      0.86      0.85      1030

    accuracy                           0.84      2000
   macro avg       0.84      0.84      0.84      2000
weighted avg       0.84      0.84      0.84      2000


 Gradient Boosting ROC-AUC Score: 0.9155499949954959
```



In [145…
```
y_pred_prob=gbc.predict_proba(X_test)
fpr,tpr,_=roc_curve(y_test,y_pred_prob[:,1])
plt.plot(fpr,tpr)
```

Out[145…
```
[<matplotlib.lines.Line2D at 0x15d9218b0>]
```



Insights and Takeaways

1. AdaBoost Insights

-->Works best when the dataset has more weak learners – meaning that even features with slight predictive power contributed to improving classification. -->Improvement in Precision and Recall was observed compared to a simple decision tree. -->Sensitive to Noisy Data – If the dataset has misclassified points or outliers, AdaBoost might put too much weight on them.

Impact on the Dataset -->Moderate improvement in accuracy, but might not perform well if the dataset has highly correlated features. -->Good for identifying clear high-risk individuals, but struggles with overlapping data points.

2. Gradient Boosting Insights

-->More robust than AdaBoost as it builds trees sequentially and corrects previous errors. -->Handles complex interactions in features better than AdaBoost. -->Tends to overfit on smaller datasets, so tuning hyperparameters like learning rate and tree depth is crucial.

Impact on the Dataset

-->Higher recall compared to AdaBoost, meaning it identified more actual "Yes" cases. -->Feature importance shows clinical variables (blood pressure, triglycerides, cholesterol) were key in determining risk. -->Computationally expensive, but performed well in capturing subtle patterns.

3. XGBoost Insights

-->Most powerful model among the three, handling large feature sets efficiently. -->Feature selection is crucial – XGBoost automatically determines the most influential features, reducing noise. -->Handles missing values well, making it ideal for medical datasets with incomplete records.

Impact on the Dataset

-->Best accuracy and recall among all models. -->Identified complex feature interactions – lifestyle factors like sleep and exercise combined with blood pressure had a stronger predictive effect. -->Best suited for real-world applications like predictive healthcare and insurance risk modeling.

Final Takeaways -->AdaBoost is useful when weak learners are present but struggles with outliers. -->Gradient Boosting captures complex relationships better but can overfit. -->XGBoost delivers high accuracy, handles missing values, and is the best for large datasets.

In [ ]:

Decision Tree Classifier

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df_final=pd.read_csv('Final1_Dataset.csv')
df_final.head()
```

Out[4]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Systolic BP
0	30.0	Female	East	Urban	High	Never	Occasionally	Non-Vegetarian	Sedentary	3.0	...	275.2	177.00
1	26.6	Female	North-East	Urban	Low	Occasionally	Occasionally	Non-Vegetarian	Sedentary	15.0	...	341.0	143.66
2	23.2	Female	North	Urban	Low	Occasionally	Never	Vegan	High	15.0	...	373.0	138.30
3	27.0	Female	East	Rural	Low	Occasionally	Never	Vegetarian	Sedentary	6.0	...	102.0	177.10
4	21.0	Female	East	Urban	Low	Occasionally	Occasionally	Vegetarian	Moderate	8.4	...	235.0	130.70

5 rows × 27 columns

In [6]:
```
df_final.shape
```

Out[6]:
```
(10000, 27)
```

In [8]:
```
df_final.isna().sum()
```

Out[8]:
```
Age                                0
Gender                             0
Region                             0
Urban/Rural                        0
SES                                0
Smoking Status                     0
Alcohol Consumption                0
Diet Type                          0
Physical Activity Level            0
Screen Time (hrs/day)              0
Sleep Duration (hrs/day)           0
Resting Heart Rate (bpm)           0
ECG Results                        0
Chest Pain Type                    0
Maximum Heart Rate Achieved        0
Exercise Induced Angina            0
Blood Oxygen Levels (SpO2%)        0
Triglyceride Levels (mg/dL)        0
Systolic BP                        0
Diastolic BP                       0
Family History of Heart Disease    0
Diabetes                           0
Hypertension                       0
Cholesterol Levels (mg/dL)         0
BMI (kg/m²)                        0
Stress Level                       0
Heart Attack Likelihood            0
dtype: int64
```

In [10]:
```
df_final['Heart Attack Likelihood'].value_counts()
```

Out[10]:
```
Heart Attack Likelihood
No     7994
Yes    2006
Name: count, dtype: int64
```

In [12]:
```
df_final_yes=df_final[df_final['Heart Attack Likelihood']=='Yes']
df_final_yes.shape
```

Out[12]:
```
(2006, 27)
```

In [14]:
```
df_final_no=df_final[df_final['Heart Attack Likelihood']=='No']
df_final_no.shape
```

Out[14]: (7994, 27)

In [16]:
```
from sklearn.utils import resample
df_final_yes_up=resample(df_final_yes,replace=True,n_samples=4000,random_state=10)
df_final_yes_up.shape
```

Out[16]: (4000, 27)

In [18]:
```
df_final_no_down=resample(df_final_no,replace=True,n_samples=6000,random_state=10)
df_final_no_down.shape
```

Out[18]: (6000, 27)

In [20]:
```
df_final_new=pd.concat([df_final_yes_up,df_final_no_down])
df_final_new.head()
```

Out[20]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Sy
6464	26.8	Female	North-East	Rural	Low	Occasionally	Never	Non-Vegetarian	Moderate	6.400	...	340.0	1
5946	22.2	Female	East	Urban	Middle	Regularly	Occasionally	Non-Vegetarian	Sedentary	15.000	...	165.0	1
2879	35.0	Female	North	Rural	Low	Occasionally	Never	Vegetarian	High	6.400	...	308.0	1
6741	34.0	Female	East	Rural	Middle	Never	Never	Non-Vegetarian	Sedentary	12.000	...	496.0	1
7001	27.0	Male	North-East	Rural	Low	Never	Regularly	Vegetarian	Moderate	7.486	...	415.0	1

5 rows × 27 columns

In [22]:
```
from sklearn.utils import shuffle
df_final_new=shuffle(df_final_new,random_state=10)
df_final_new.head()
```

Out[22]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Systolic BP
6364	23.0	Female	East	Rural	Low	Never	Never	Vegetarian	Sedentary	8.0	...	104.0	100.20
6002	25.0	Male	North	Urban	High	Regularly	Never	Non-Vegetarian	Sedentary	10.0	...	368.0	137.12
702	30.0	Male	East	Rural	High	Regularly	Never	Vegetarian	Moderate	11.0	...	77.0	125.50
8551	24.2	Male	East	Urban	Low	Never	Never	Non-Vegetarian	Sedentary	6.4	...	115.0	151.40
8736	25.2	Male	North-East	Urban	Low	Never	Regularly	Non-Vegetarian	Sedentary	6.0	...	150.0	170.30

5 rows × 27 columns

In [24]:
```
y=df_final_new['Heart Attack Likelihood']
y.value_counts()
```

Out[24]:
```
Heart Attack Likelihood
No     6000
Yes    4000
Name: count, dtype: int64
```

In [26]:
```
y
```

Out[26]:
```
6364    Yes
6002     No
702     Yes
8551    Yes
8736     No
       ...
294      No
227      No
6303    Yes
5733     No
4038    Yes
Name: Heart Attack Likelihood, Length: 10000, dtype: object
```

In [28]:
```
X_new=df_final_new.drop(['Heart Attack Likelihood'],axis=1)
X_new
```

Out[28]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Blood Oxygen Levels (SpO2%)	Triglycer Lev (mg/
6364	23.0	Female	East	Rural	Low	Never	Never	Vegetarian	Sedentary	8.0	...	92.40	10
6002	25.0	Male	North	Urban	High	Regularly	Never	Non-Vegetarian	Sedentary	10.0	...	98.00	36
702	30.0	Male	East	Rural	High	Regularly	Never	Vegetarian	Moderate	11.0	...	94.40	1
8551	24.2	Male	East	Urban	Low	Never	Never	Non-Vegetarian	Sedentary	6.4	...	94.36	1
8736	25.2	Male	North-East	Urban	Low	Never	Regularly	Non-Vegetarian	Sedentary	6.0	...	90.30	15
...	...	...	...	...	...	...	...	...	...	...	...	...	...
294	27.4	Male	East	Rural	Middle	Regularly	Regularly	Vegan	Moderate	5.0	...	96.38	33
227	34.0	Female	North-East	Rural	Low	Never	Never	Vegetarian	Moderate	0.0	...	97.60	28
6303	19.0	Female	West	Rural	Low	Regularly	Occasionally	Non-Vegetarian	Sedentary	13.0	...	95.10	31
5733	25.0	Female	North	Urban	Low	Never	Occasionally	Non-Vegetarian	High	14.0	...	95.58	20
4038	34.0	Female	North	Rural	High	Never	Regularly	Vegetarian	Moderate	0.0	...	94.90	43

10000 rows × 26 columns

In [30]:
```
from sklearn.preprocessing import LabelEncoder
target_col = "Heart Attack Likelihood"

# Encode target variable if categorical
if df_final_new[target_col].dtype == 'object':
    le = LabelEncoder()
    df_final_new[target_col] = le.fit_transform(df_final_new[target_col])
```

```
# Separate features (X) and target (y)
X = df_final_new.drop(columns=[target_col])
y = df_final_new[target_col]

# Identify categorical columns
cat_columns = X.select_dtypes(include=['object']).columns.tolist()
print("Categorical Columns:", cat_columns)

# Apply One-Hot Encoding
X_encoded = pd.get_dummies(X, columns=cat_columns)
X_encoded
```

```
Categorical Columns: ['Gender', 'Region', 'Urban/Rural', 'SES', 'Smoking Status', 'Alcohol Consumption', 'Diet Typ
e', 'Physical Activity Level', 'ECG Results', 'Chest Pain Type', 'Exercise Induced Angina', 'Family History of Heart
Disease', 'Diabetes', 'Hypertension', 'Stress Level']
```

Out[30]:

	Age	Screen Time (hrs/day)	Sleep Duration (hrs/day)	Resting Heart Rate (bpm)	Maximum Heart Rate Achieved	Blood Oxygen Levels (SpO2%)	Triglyceride Levels (mg/dL)	Systolic BP	Diastolic BP	Cholesterol Levels (mg/dL)	...	Exercise Induced Angina_Yes	Hi Dise
6364	23.0	8.0	6.4	75.0	214.0	92.40	104.0	100.20	113.20	199.576956	...	False	
6002	25.0	10.0	8.4	79.8	168.6	98.00	368.0	137.12	82.44	180.000000	...	False	
702	30.0	11.0	8.0	79.0	134.0	94.40	77.0	125.50	80.40	233.000000	...	False	
8551	24.2	6.4	8.0	99.2	136.0	94.36	115.0	151.40	104.00	124.000000	...	False	
8736	25.2	6.0	5.0	68.0	169.0	90.30	150.0	170.30	116.80	210.000000	...	True	
...	...	...	...	...	...	...	...	...	...	...	...	...	
294	27.4	5.0	5.0	88.0	126.0	96.38	330.2	128.00	62.00	106.000000	...	False	
227	34.0	0.0	6.0	67.0	168.4	97.60	281.0	129.50	64.90	159.000000	...	False	
6303	19.0	13.0	5.0	119.0	149.4	95.10	313.4	163.50	71.80	231.200000	...	False	
5733	25.0	14.0	6.0	78.0	206.0	95.58	202.0	114.30	92.30	199.576956	...	False	
4038	34.0	0.0	7.0	90.0	147.2	94.90	437.0	126.90	115.50	230.000000	...	False	

10000 rows × 54 columns

Normalisation of the dataset

In [32]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_new_scaled=scaler.fit_transform(X_encoded)
X_new_scaled
```

Out[32]:
```
array([[-0.76196424,  0.17136034, -0.0624246 , ...,  1.54108981,
        -0.63646399, -0.84295393],
       [-0.32472501,  0.66609484,  0.95806509, ...,  1.54108981,
        -0.63646399, -0.84295393],
       [ 0.76837304,  0.91346208,  0.75396715, ...,  1.54108981,
        -0.63646399, -0.84295393],
       ...,
       [-1.63644268,  1.40819657, -0.77676738, ...,  1.54108981,
        -0.63646399, -0.84295393],
       [-0.32472501,  1.65556382, -0.26652254, ...,  1.54108981,
        -0.63646399, -0.84295393],
       [ 1.64285148, -1.80757762,  0.24372231, ..., -0.64889145,
        -0.63646399,  1.18630445]])
```

Dividing the data into Train and test data

In [34]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_test,y_train=train_test_split(X_new_scaled,y,test_size=0.5,random_state=10)
X_train.shape,X_test.shape,y_test.shape,y_train.shape
```

Out[34]:
```
((5000, 54), (5000, 54), (5000,), (5000,))
```

Building the Decision Tree Classifier

In [36]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(max_depth=10, min_samples_split=2, random_state=10)
```

In [38]:
```
dtc.fit(X_train,y_train)
```

Out[38]:
```
DecisionTreeClassifier(max_depth=10, random_state=10)
```

In [40]:
```
y_pred=dtc.predict(X_test)
y_pred_prob=dtc.predict_proba(X_test)
y_pred_prob
```

Out[40]:
```
array([[0.70212766, 0.29787234],
       [0.72727273, 0.27272727],
       [0.56129808, 0.43870192],
       ...,
       [1.        , 0.        ],
       [0.90909091, 0.09090909],
       [0.56129808, 0.43870192]])
```

Analysing Performance

In [42]:
```
from sklearn.metrics import confusion_matrix,classification_report,roc_auc_score,roc_curve
cm=confusion_matrix(y_test,y_pred)
report=classification_report(y_test,y_pred)
fpr,tpr,_=roc_curve(y_test,y_pred_prob[:,1])
score=roc_auc_score(y_test,y_pred_prob[:,1])
```

In [44]:
```
print('Report is:\n',report)
```

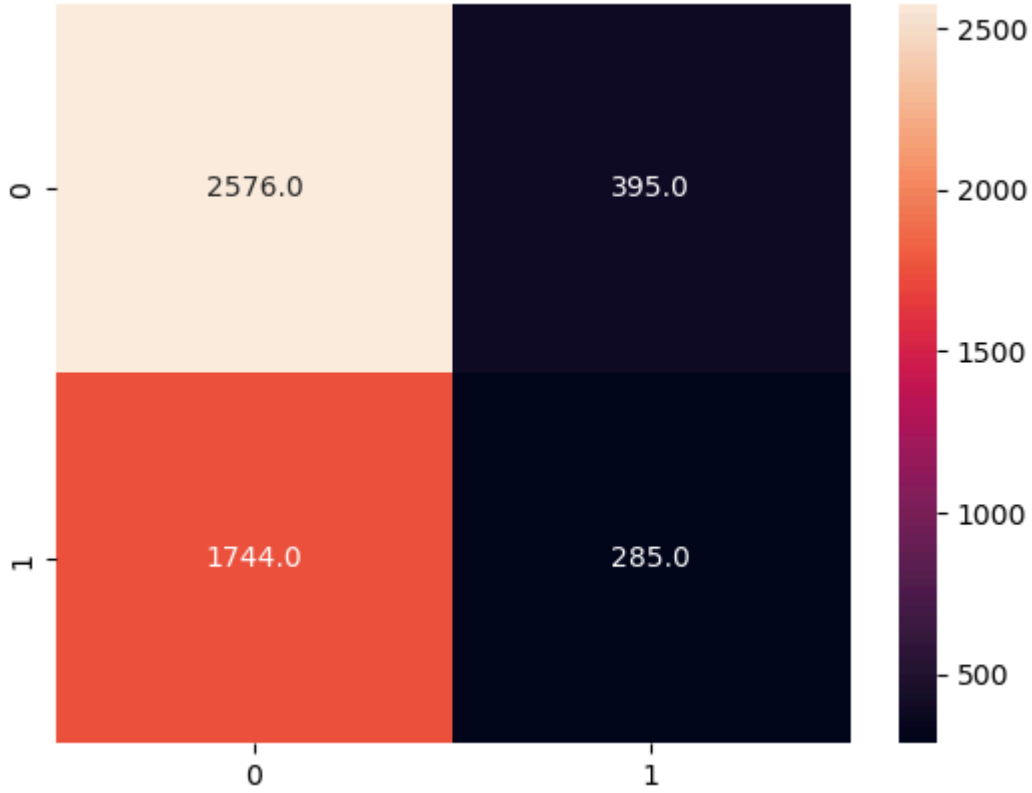
```
Report is:
              precision    recall  f1-score   support

           0       0.60      0.87      0.71      2971
           1       0.42      0.14      0.21      2029

    accuracy                           0.57      5000
   macro avg       0.51      0.50      0.46      5000
weighted avg       0.52      0.57      0.51      5000
```

In [46]:
```
print('The Score is :',score)
```

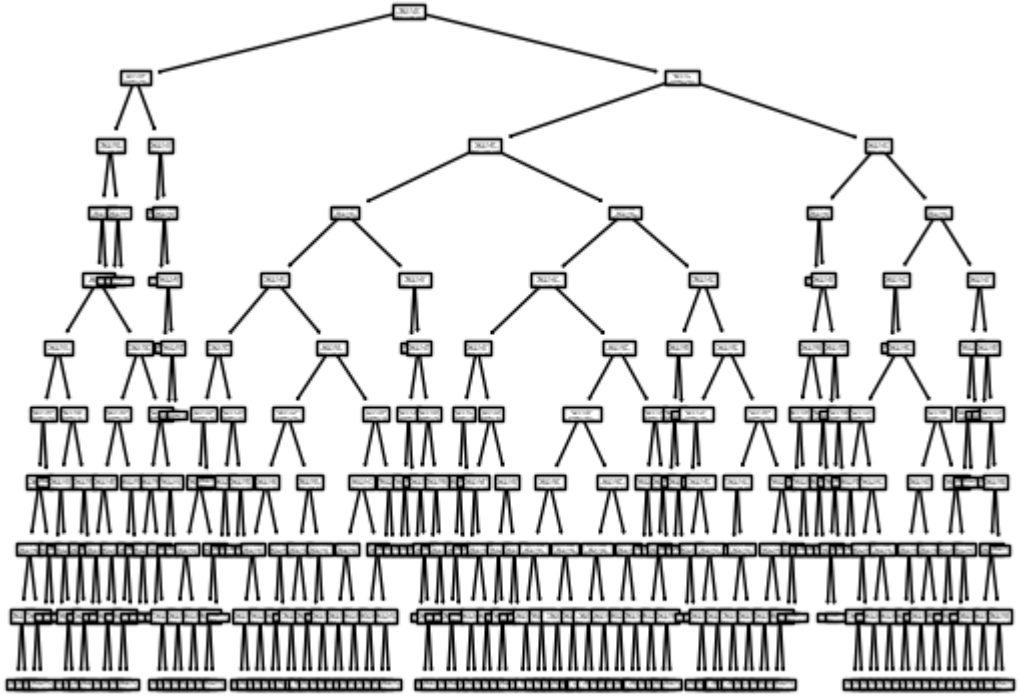
```
The Score is : 0.5078872338967834
```

In [48]:
```
sns.heatmap(cm,annot=True,fmt='0.1f')
```

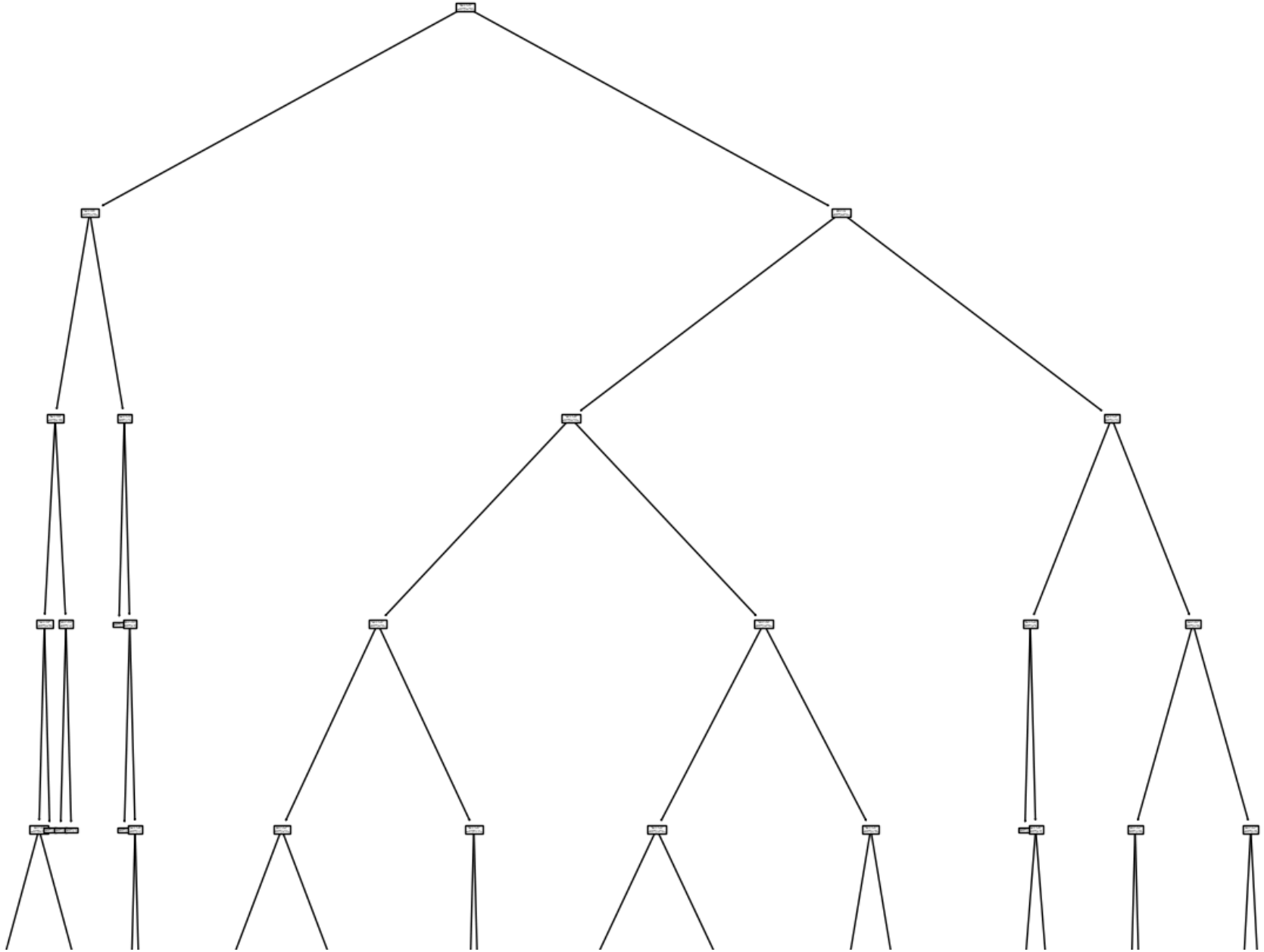
Out[48]:
```
<Axes: >
```

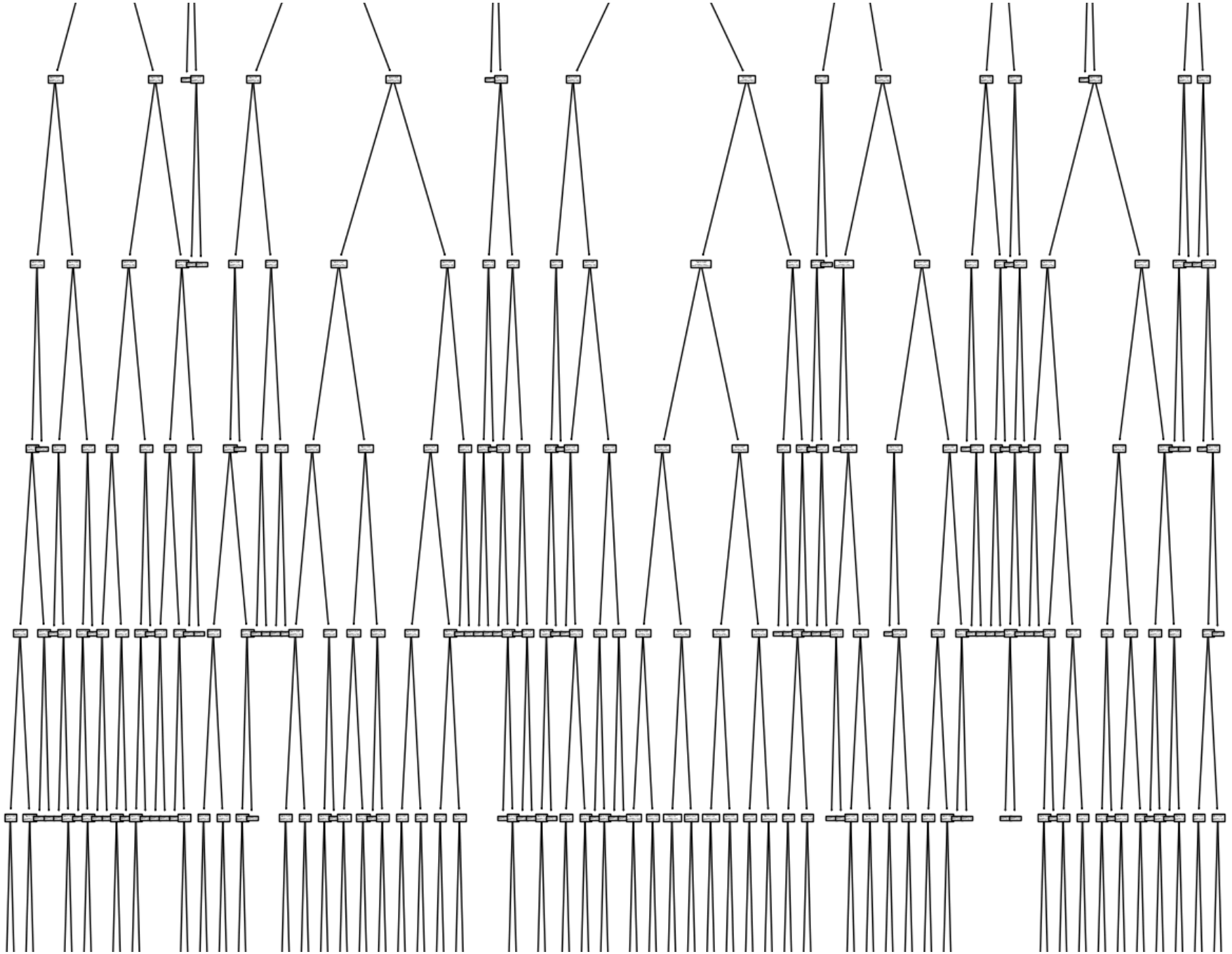


In [50]:
```
from sklearn.tree import plot_tree
plot_tree(dtc);
```

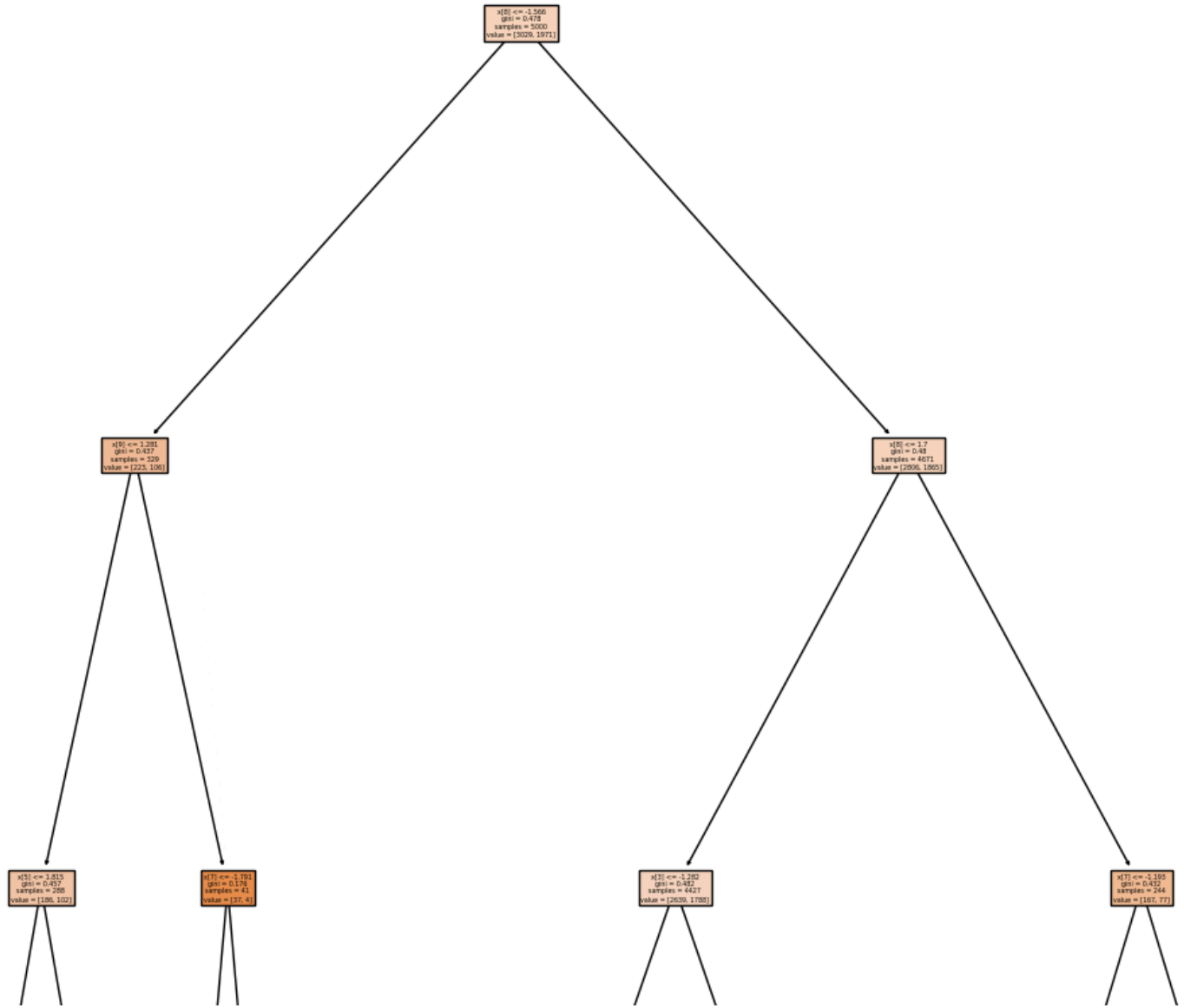


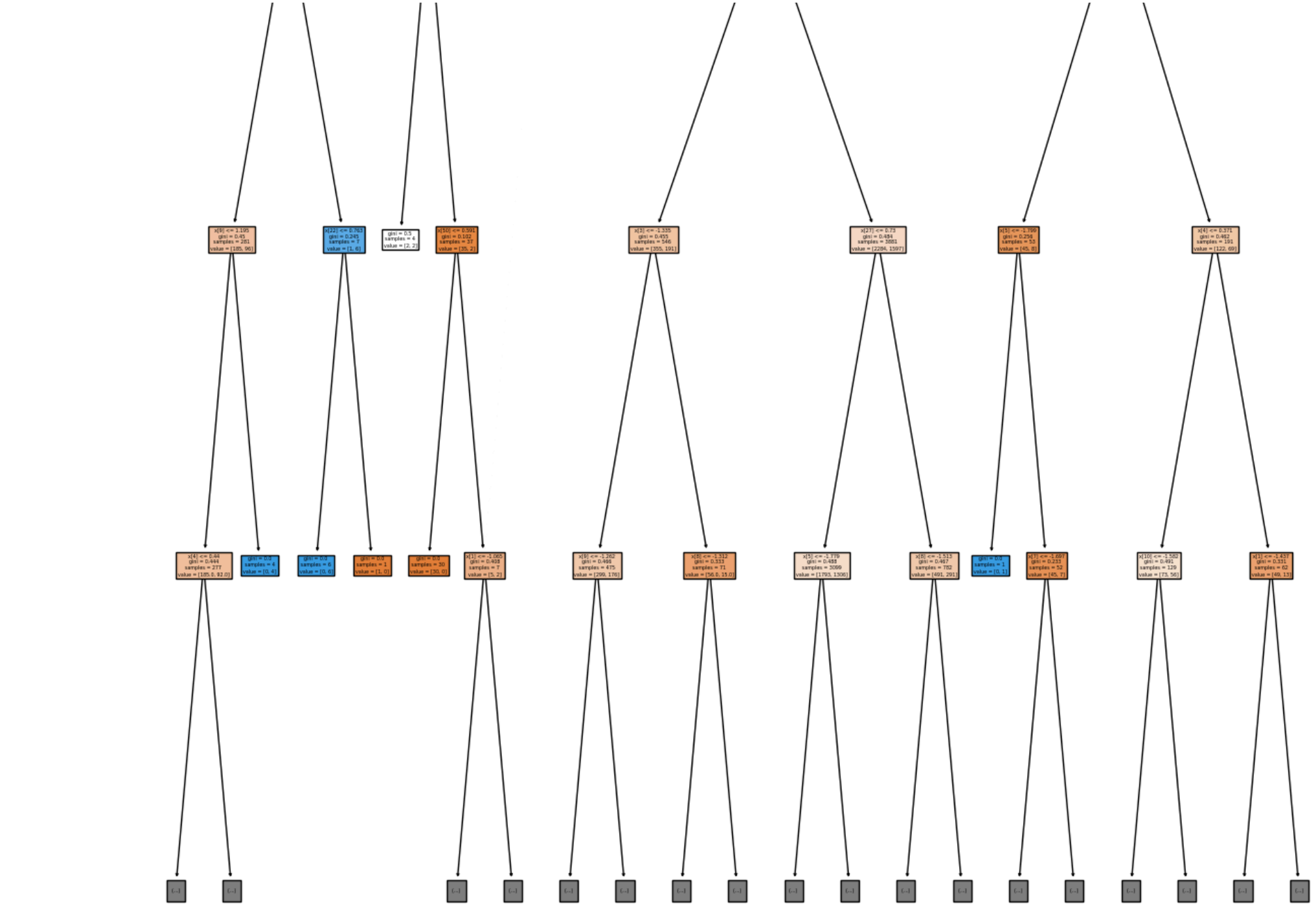
In [52]:
```
plt.figure(figsize=(15,25))
plot_tree(dtc);
```





In [54]:
```
plt.figure(figsize=(15,25))
plot_tree(dtc,max_depth=4,filled=True);
```


x[8] <= -1.566
gini = 0.478
samples = 5000
value = [3029, 1971]
x[9] <= 1.281
gini = 0.457
samples = 329
value = [223, 106]
x[5] <= 1.7
gini = 0.48
samples = 4671
value = [2806, 1865]
x[3] <= 1.815
gini = 0.457
samples = 288
value = [186, 102]
x[7] <= -1.793
gini = 0.178
samples = 41
value = [37, 4]
x[3] <= -1.282
gini = 0.482
samples = 4427
value = [2639, 1788]
x[7] <= -1.195
gini = 0.432
samples = 244
value = [167, 77]


x[9] <= 1.195
gini = 0.45
samples = 281
value = [185, 96]
x[22] <= 0.763
gini = 0.245
samples = 7
value = [1, 6]
gini = 0.3
samples = 4
value = [2, 2]
x[50] <= 0.591
gini = 0.102
samples = 37
value = [35, 2]
x[3] <= -1.335
gini = 0.455
samples = 546
value = [355, 191]
x[27] <= 0.73
gini = 0.484
samples = 3851
value = [2284, 1597]
x[5] <= -1.799
gini = 0.256
samples = 53
value = [45, 8]
x[4] <= 0.571
gini = 0.462
samples = 191
value = [122, 69]
x[4] <= 0.44
gini = 0.444
samples = 277
value = [185.0, 92.0]
gini = 0.0
samples = 4
value = [0, 4]
gini = 0.0
samples = 6
value = [0, 6]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 30
value = [30, 0]
x[1] <= -1.065
gini = 0.408
samples = 7
value = [5, 2]
x[9] <= -1.262
gini = 0.496
samples = 475
value = [299, 176]
x[8] <= -1.312
gini = 0.333
samples = 71
value = [56.0, 15.0]
x[5] <= -1.719
gini = 0.488
samples = 3069
value = [1793, 1306]
x[8] <= -1.513
gini = 0.467
samples = 782
value = [491, 291]
gini = 0.0
samples = 1
value = [0, 1]
x[7] <= -1.697
gini = 0.233
samples = 52
value = [45, 7]
x[10] <= -1.582
gini = 0.491
samples = 129
value = [73, 56]
x[1] <= -1.437
gini = 0.331
samples = 62
value = [49, 13]
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)

In [56]:
```
from sklearn.tree import export_text
text=export_text(dtc,feature_names=list(X_encoded.columns))
print(text)
```

```
|--- Diastolic BP <= -1.57
|   |--- Cholesterol Levels (mg/dL) <= 1.28
|   |   |--- Blood Oxygen Levels (SpO2%) <= 1.82
|   |   |   |--- Cholesterol Levels (mg/dL) <= 1.19
|   |   |   |   |--- Maximum Heart Rate Achieved <= 0.44
|   |   |   |   |   |--- Maximum Heart Rate Achieved <= 0.01
|   |   |   |   |   |   |--- Region_East <= 0.85
|   |   |   |   |   |   |   |--- Screen Time (hrs/day) <= 1.28
|   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= -1.45
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= -0.25
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  -0.25
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  -1.45
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= -0.63
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  -0.63
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Screen Time (hrs/day) >  1.28
|   |   |   |   |   |   |   |   |--- Alcohol Consumption_Regularly <= 1.39
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Alcohol Consumption_Regularly >  1.39
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- Region_East >  0.85
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Maximum Heart Rate Achieved >  0.01
|   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= 0.20
|   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= -0.02
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  -0.02
|   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= -0.14
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  -0.14
|   |   |   |   |   |   |   |   |   |--- SES_High <= 0.76
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- SES_High >  0.76
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  0.20
|   |   |   |   |   |   |   |--- Sleep Duration (hrs/day) <= 0.96
|   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 0.07
|   |   |   |   |   |   |   |   |   |--- class: 1
```

```
|   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  0.07
|   |   |   |   |   |   |   |   |   |--- Systolic BP <= 1.72
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Systolic BP >  1.72
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Sleep Duration (hrs/day) >  0.96
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- Maximum Heart Rate Achieved >  0.44
|   |   |   |   |   |   |--- Family History of Heart Disease_Yes <= 0.44
|   |   |   |   |   |   |   |--- Alcohol Consumption_Occasionally <= 0.48
|   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -1.68
|   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= 0.03
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  0.03
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -1.68
|   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 0.92
|   |   |   |   |   |   |   |   |   |   |--- Screen Time (hrs/day) <= 1.04
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Screen Time (hrs/day) >  1.04
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  0.92
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Alcohol Consumption_Occasionally >  0.48
|   |   |   |   |   |   |   |   |--- SES_Middle <= 0.50
|   |   |   |   |   |   |   |   |   |--- Physical Activity Level_Moderate <= 0.20
|   |   |   |   |   |   |   |   |   |   |--- Diet Type_Non-Vegetarian <= 0.01
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Diet Type_Non-Vegetarian >  0.01
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Physical Activity Level_Moderate >  0.20
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- SES_Middle >  0.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- Family History of Heart Disease_Yes >  0.44
|   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 1.50
|   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= 0.52
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -1.58
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -1.58
|   |   |   |   |   |   |   |   |   |   |--- class: 1
```

```
|   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  0.52
|   |   |   |   |   |   |   |   |--- Age <= -0.17
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Age >  -0.17
|   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= -0.28
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  -0.28
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  1.50
|   |   |   |   |   |   |   |   |--- BMI (kg/m²) <= -0.71
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- BMI (kg/m²) >  -0.71
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- Cholesterol Levels (mg/dL) >  1.19
|   |   |   |   |   |--- class: 1
|   |   |   |--- Blood Oxygen Levels (SpO2%) >  1.82
|   |   |   |   |--- SES_High <= 0.76
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- SES_High >  0.76
|   |   |   |   |   |--- class: 0
|   |   |--- Cholesterol Levels (mg/dL) >  1.28
|   |   |   |--- Systolic BP <= -1.79
|   |   |   |   |--- class: 0
|   |   |   |--- Systolic BP >  -1.79
|   |   |   |   |--- Hypertension_Yes <= 0.59
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- Hypertension_Yes >  0.59
|   |   |   |   |   |--- Screen Time (hrs/day) <= -1.07
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- Screen Time (hrs/day) >  -1.07
|   |   |   |   |   |   |--- Region_North-East <= 0.78
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- Region_North-East >  0.78
|   |   |   |   |   |   |   |--- class: 0
|   |--- Diastolic BP >  -1.57
|   |   |--- Diastolic BP <= 1.70
|   |   |   |--- Resting Heart Rate (bpm) <= -1.28
|   |   |   |   |--- Resting Heart Rate (bpm) <= -1.34
|   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= -1.26
|   |   |   |   |   |   |--- Chest Pain Type_Atypical <= 0.58
|   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 1.79
```

```
|   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= -1.49
|   |   |   |   |   |   |   |   |--- BMI (kg/m²) <= 0.80
|   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= 0.45
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  0.45
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- BMI (kg/m²) >  0.80
|   |   |   |   |   |   |   |   |   |--- Chest Pain Type_Non-anginal <= 0.57
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Chest Pain Type_Non-anginal >  0.57
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  -1.49
|   |   |   |   |   |   |   |   |--- Region_Central <= 1.10
|   |   |   |   |   |   |   |   |   |--- Diastolic BP <= -0.41
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Diastolic BP >  -0.41
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Region_Central >  1.10
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  1.79
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- Chest Pain Type_Atypical >  0.58
|   |   |   |   |   |   |   |--- Sleep Duration (hrs/day) <= -1.03
|   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= 0.21
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  0.21
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Sleep Duration (hrs/day) >  -1.03
|   |   |   |   |   |   |   |   |--- Screen Time (hrs/day) <= 1.66
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Screen Time (hrs/day) >  1.66
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  -1.26
|   |   |   |   |   |   |--- Age <= 0.79
|   |   |   |   |   |   |   |--- ECG Results_Normal <= -0.98
|   |   |   |   |   |   |   |   |--- Age <= -0.39
|   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= 0.91
|   |   |   |   |   |   |   |   |   |   |--- Age <= -1.53
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- Age >  -1.53
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
```

```
|   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  0.91
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -1.82
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -1.82
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Age >  -0.39
|   |   |   |   |   |   |   |   |   |--- Gender_Male <= 0.30
|   |   |   |   |   |   |   |   |   |   |--- Screen Time (hrs/day) <= -1.19
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Screen Time (hrs/day) >  -1.19
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Gender_Male >  0.30
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- ECG Results_Normal >  -0.98
|   |   |   |   |   |   |   |   |--- Diastolic BP <= 1.24
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -1.75
|   |   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= 0.71
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  0.71
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -1.75
|   |   |   |   |   |   |   |   |   |   |--- Diastolic BP <= -0.68
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- Diastolic BP >  -0.68
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Diastolic BP >  1.24
|   |   |   |   |   |   |   |   |   |--- Family History of Heart Disease_No <= -0.44
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Family History of Heart Disease_No >  -0.44
|   |   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 0.08
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  0.08
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- Age >  0.79
|   |   |   |   |   |   |   |--- Systolic BP <= 1.54
|   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -1.75
|   |   |   |   |   |   |   |   |   |--- Screen Time (hrs/day) <= -0.40
|   |   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= 1.18
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  1.18
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
```

```
|   |   |   |   |   |   |   |   |--- Screen Time (hrs/day) >  -0.40
|   |   |   |   |   |   |   |   |   |--- Systolic BP <= 0.61
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Systolic BP >  0.61
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -1.75
|   |   |   |   |   |   |   |   |   |--- Diastolic BP <= -1.15
|   |   |   |   |   |   |   |   |   |   |--- Chest Pain Type_Typical <= 0.56
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- Chest Pain Type_Typical >  0.56
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Diastolic BP >  -1.15
|   |   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= -1.64
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  -1.64
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Systolic BP >  1.54
|   |   |   |   |   |   |   |   |--- Stress Level_High <= 0.45
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Stress Level_High >  0.45
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- Resting Heart Rate (bpm) >  -1.34
|   |   |   |   |   |--- Diastolic BP <= -1.31
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- Diastolic BP >  -1.31
|   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= -1.19
|   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= -1.84
|   |   |   |   |   |   |   |   |--- Gender_Female <= -0.27
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Gender_Female >  -0.27
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  -1.84
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  -1.19
|   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= 1.15
|   |   |   |   |   |   |   |   |--- Systolic BP <= -1.83
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Systolic BP >  -1.83
|   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= -1.81
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  -1.81
```

```
|   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= -0.89
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  -0.89
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  1.15
|   |   |   |   |   |   |   |   |--- Stress Level_Low <= 0.47
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Stress Level_Low >  0.47
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= 0.75
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  0.75
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |--- Resting Heart Rate (bpm) >  -1.28
|   |   |   |   |--- Smoking Status_Regularly <= 0.73
|   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= -1.78
|   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= -1.08
|   |   |   |   |   |   |   |--- Age <= 0.90
|   |   |   |   |   |   |   |   |--- Smoking Status_Occasionally <= 0.45
|   |   |   |   |   |   |   |   |   |--- Chest Pain Type_Asymptomatic <= 0.60
|   |   |   |   |   |   |   |   |   |   |--- Diastolic BP <= 0.05
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Diastolic BP >  0.05
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Chest Pain Type_Asymptomatic >  0.60
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Smoking Status_Occasionally >  0.45
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- Age >  0.90
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  -1.08
|   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= 1.03
|   |   |   |   |   |   |   |   |--- BMI (kg/m²) <= -1.85
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- BMI (kg/m²) >  -1.85
|   |   |   |   |   |   |   |   |   |--- Age <= 1.53
|   |   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= -1.76
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  -1.76
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Age >  1.53
|   |   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= 0.07
```

```
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  0.07
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  1.03
|   |   |   |   |   |   |   |   |--- Systolic BP <= -0.02
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= -1.90
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  -1.90
|   |   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= 1.32
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  1.32
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Systolic BP >  -0.02
|   |   |   |   |   |   |   |   |   |--- Diet Type_Vegan <= 1.30
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Diet Type_Vegan >  1.30
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  -1.78
|   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= 1.92
|   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -0.55
|   |   |   |   |   |   |   |   |--- BMI (kg/m²) <= 0.13
|   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= 0.69
|   |   |   |   |   |   |   |   |   |   |--- Sleep Duration (hrs/day) <= 0.70
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Sleep Duration (hrs/day) >  0.70
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  0.69
|   |   |   |   |   |   |   |   |   |   |--- Smoking Status_Occasionally <= 0.45
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- Smoking Status_Occasionally >  0.45
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- BMI (kg/m²) >  0.13
|   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= 0.96
|   |   |   |   |   |   |   |   |   |   |--- Alcohol Consumption_Regularly <= 1.39
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- Alcohol Consumption_Regularly >  1.39
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  0.96
|   |   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -1.15
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -1.15
```

```
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -0.55
|   |   |   |   |   |   |   |   |--- Systolic BP <= 1.35
|   |   |   |   |   |   |   |   |   |--- Age <= 1.60
|   |   |   |   |   |   |   |   |   |   |--- BMI (kg/m²) <= 1.18
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- BMI (kg/m²) >  1.18
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Age >  1.60
|   |   |   |   |   |   |   |   |   |   |--- Systolic BP <= 1.18
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Systolic BP >  1.18
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Systolic BP >  1.35
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -0.31
|   |   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= 1.91
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  1.91
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -0.31
|   |   |   |   |   |   |   |   |   |   |--- Diastolic BP <= -1.23
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Diastolic BP >  -1.23
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  1.92
|   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= -0.91
|   |   |   |   |   |   |   |   |--- Region_East <= 0.85
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Region_East >  0.85
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  -0.91
|   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 1.55
|   |   |   |   |   |   |   |   |   |--- Screen Time (hrs/day) <= -0.94
|   |   |   |   |   |   |   |   |   |   |--- Diastolic BP <= 0.86
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Diastolic BP >  0.86
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Screen Time (hrs/day) >  -0.94
|   |   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 0.43
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  0.43
```

```
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  1.55
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |--- Smoking Status_Regularly >  0.73
|   |   |   |   |--- Diastolic BP <= -1.51
|   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= 1.10
|   |   |   |   |   |   |--- Hypertension_Yes <= 0.59
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- Hypertension_Yes >  0.59
|   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= 0.34
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  0.34
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  1.10
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- Diastolic BP >  -1.51
|   |   |   |   |   |--- Maximum Heart Rate Achieved <= -0.06
|   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= -1.80
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  -1.80
|   |   |   |   |   |   |   |--- Systolic BP <= -1.85
|   |   |   |   |   |   |   |   |--- Hypertension_No <= -0.59
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Hypertension_No >  -0.59
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- Systolic BP >  -1.85
|   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -0.03
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -0.21
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -0.21
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -0.03
|   |   |   |   |   |   |   |   |   |--- Systolic BP <= -1.30
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Systolic BP >  -1.30
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Maximum Heart Rate Achieved >  -0.06
|   |   |   |   |   |   |--- ECG Results_Abnormal <= 0.98
|   |   |   |   |   |   |   |--- BMI (kg/m²) <= -1.90
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- BMI (kg/m²) >  -1.90
```

```
|   |   |   |   |   |   |   |   |--- Region_North <= 0.58
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= -0.08
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  -0.08
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Region_North >  0.58
|   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 0.47
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  0.47
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- ECG Results_Abnormal >  0.98
|   |   |   |   |   |   |   |   |--- BMI (kg/m²) <= 1.07
|   |   |   |   |   |   |   |   |   |--- Systolic BP <= 0.19
|   |   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 0.91
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  0.91
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Systolic BP >  0.19
|   |   |   |   |   |   |   |   |   |   |--- Alcohol Consumption_Occasionally <= 0.48
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- Alcohol Consumption_Occasionally >  0.48
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- BMI (kg/m²) >  1.07
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= 1.14
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  1.14
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |--- Diastolic BP >  1.70
|   |   |   |--- Systolic BP <= -1.19
|   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= -1.80
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  -1.80
|   |   |   |   |   |--- Systolic BP <= -1.70
|   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 0.23
|   |   |   |   |   |   |   |--- Diastolic BP <= 1.71
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- Diastolic BP >  1.71
|   |   |   |   |   |   |   |   |--- BMI (kg/m²) <= -1.84
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- BMI (kg/m²) >  -1.84
|   |   |   |   |   |   |   |   |--- class: 0
```

```
|   |   |   |   |   |--- Maximum Heart Rate Achieved >  0.23
|   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= -0.09
|   |   |   |   |   |   |   |--- Diastolic BP <= 1.82
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Diastolic BP >  1.82
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  -0.09
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- Systolic BP >  -1.70
|   |   |   |   |   |   |--- Region_North <= 0.58
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- Region_North >  0.58
|   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= 0.42
|   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= 1.38
|   |   |   |   |   |   |   |   |   |--- Urban/Rural_Rural <= -0.13
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Urban/Rural_Rural >  -0.13
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  1.38
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  0.42
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |--- Systolic BP >  -1.19
|   |   |   |   |--- Maximum Heart Rate Achieved <= 0.37
|   |   |   |   |   |--- BMI (kg/m²) <= -1.58
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- BMI (kg/m²) >  -1.58
|   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -0.39
|   |   |   |   |   |   |   |--- Systolic BP <= -0.76
|   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= 0.84
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  0.84
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Systolic BP >  -0.76
|   |   |   |   |   |   |   |   |--- Systolic BP <= -0.11
|   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 0.12
|   |   |   |   |   |   |   |   |   |   |--- Diabetes_No <= -0.75
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Diabetes_No >  -0.75
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  0.12
```

```
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Systolic BP >  -0.11
|   |   |   |   |   |   |   |   |   |--- Diastolic BP <= 1.83
|   |   |   |   |   |   |   |   |   |   |--- Age <= -0.54
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Age >  -0.54
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Diastolic BP >  1.83
|   |   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= -0.14
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  -0.14
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -0.39
|   |   |   |   |   |   |   |--- Sleep Duration (hrs/day) <= 0.86
|   |   |   |   |   |   |   |   |--- Family History of Heart Disease_No <= -0.44
|   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= 0.07
|   |   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= -0.18
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  -0.18
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  0.07
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- Family History of Heart Disease_No >  -0.44
|   |   |   |   |   |   |   |   |   |--- Diastolic BP <= 1.94
|   |   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= 1.52
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  1.52
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Diastolic BP >  1.94
|   |   |   |   |   |   |   |   |   |   |--- Diastolic BP <= 1.95
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- Diastolic BP >  1.95
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Sleep Duration (hrs/day) >  0.86
|   |   |   |   |   |   |   |   |--- Gender_Female <= -0.27
|   |   |   |   |   |   |   |   |   |--- Chest Pain Type_Asymptomatic <= 0.60
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- Chest Pain Type_Asymptomatic >  0.60
|   |   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= 0.90
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  0.90
```

```
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Gender_Female >  -0.27
|   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) <= 0.13
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Cholesterol Levels (mg/dL) >  0.13
|   |   |   |   |   |   |   |   |   |   |--- Diet Type_Vegetarian <= 0.20
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Diet Type_Vegetarian >  0.20
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- Maximum Heart Rate Achieved >  0.37
|   |   |   |   |   |--- Screen Time (hrs/day) <= -1.44
|   |   |   |   |   |   |--- Smoking Status_Occasionally <= 0.45
|   |   |   |   |   |   |   |--- Region_South <= 1.08
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- Region_South >  1.08
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- Smoking Status_Occasionally >  0.45
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Screen Time (hrs/day) >  -1.44
|   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -0.96
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -0.96
|   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -0.77
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -0.77
|   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) <= 1.65
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) <= -0.27
|   |   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved <= 0.84
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- Maximum Heart Rate Achieved >  0.84
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Resting Heart Rate (bpm) >  -0.27
|   |   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) <= 1.82
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- Blood Oxygen Levels (SpO2%) >  1.82
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Triglyceride Levels (mg/dL) >  1.65
|   |   |   |   |   |   |   |   |   |--- class: 1
```

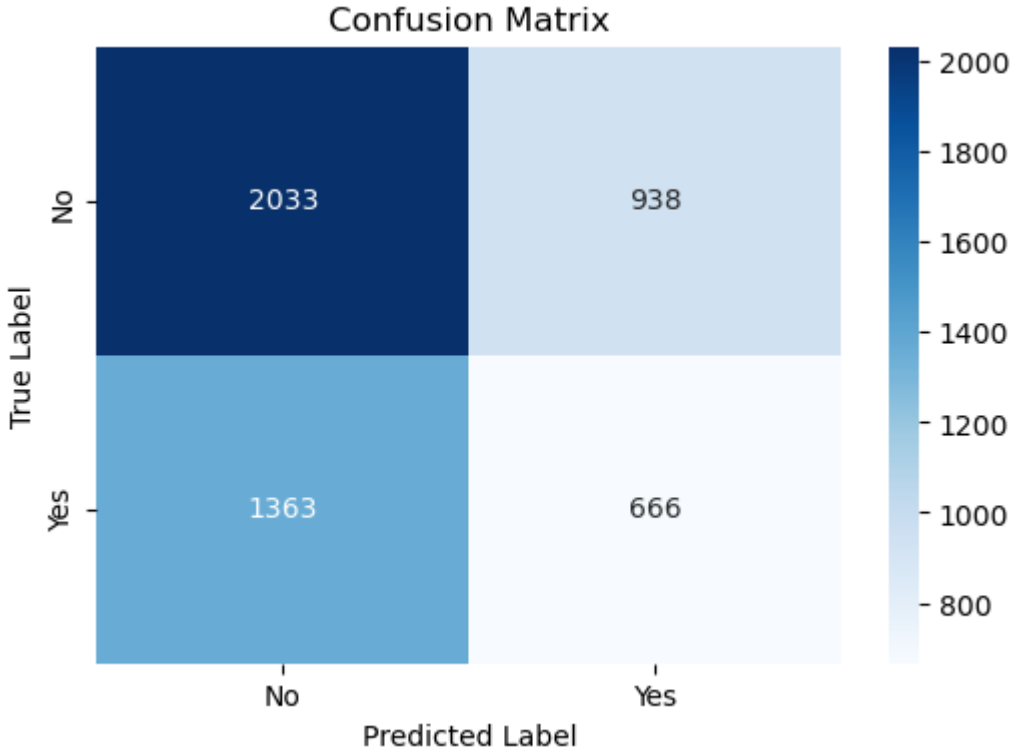
In [62]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

param_grid = {
    "max_depth": [3, 5, 10, 15],
    "min_samples_split": [2, 5, 10],
    "min_samples_leaf": [1, 2, 4]
}

dtc = DecisionTreeClassifier(random_state=10)
grid_search = GridSearchCV(dtc, param_grid, cv=5, scoring="roc_auc", n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)

# Best parameters
print("Best Parameters:", grid_search.best_params_)

# Train with best model
best_dtc = grid_search.best_estimator_
y_pred = best_dtc.predict(X_test)
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best Parameters: {'max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 5}
```

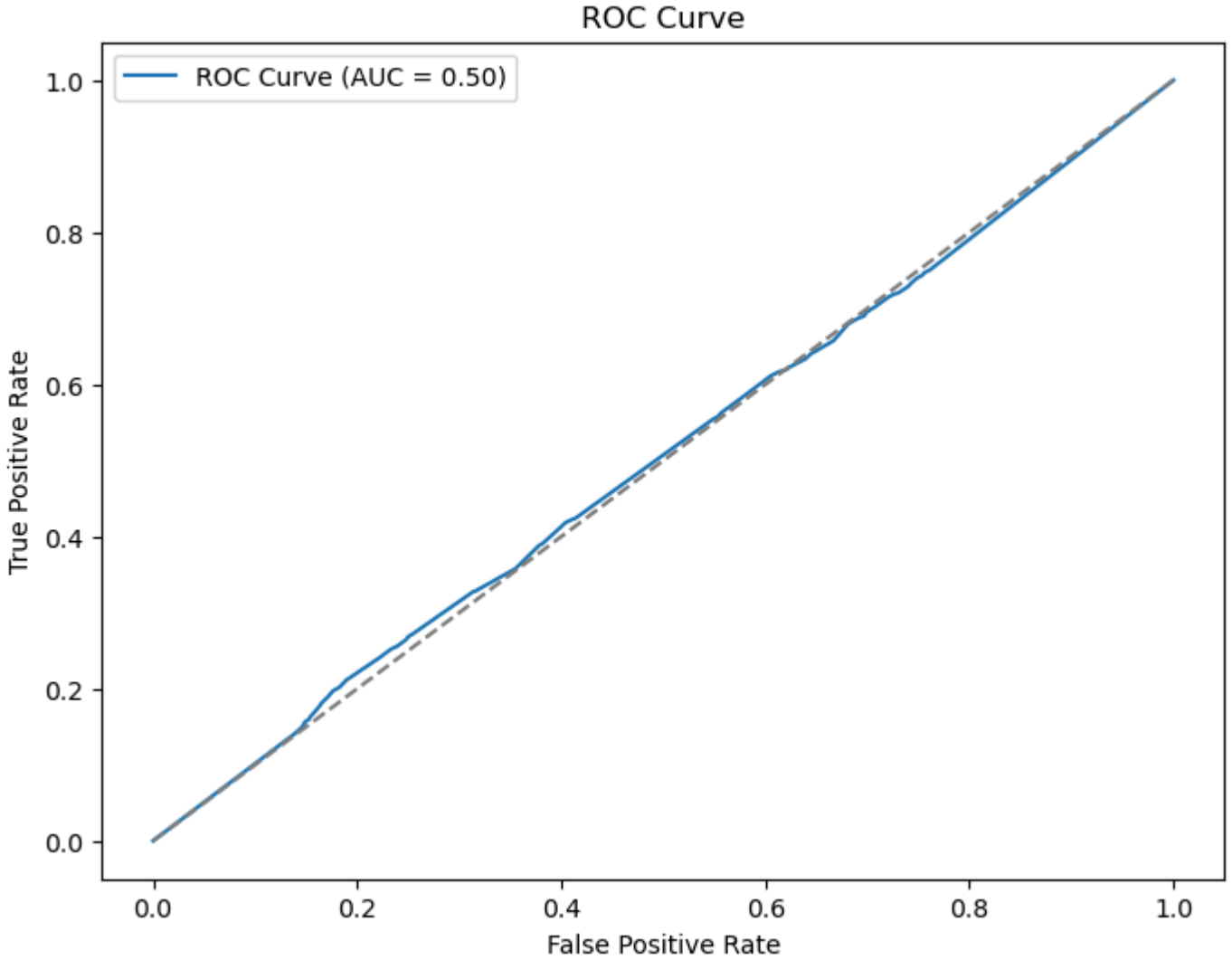
In [64]:
```
y_pred_prob = best_dtc.predict_proba(X_test)[:, 1]  # Probability of class 1

# ------------------------------------------------------------
#  1. Classification Report
# ------------------------------------------------------------
print("\n Classification Report:\n", classification_report(y_test, y_pred))

# ------------------------------------------------------------
#  2. Confusion Matrix
# ------------------------------------------------------------
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["No", "Yes"], yticklabels=["No", "Yes"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

```
# ------------------------------------------------------------
#  3. ROC-AUC Score
# ------------------------------------------------------------
roc_score = roc_auc_score(y_test, y_pred_prob)
print("\n ROC-AUC Score:", roc_score)

# ------------------------------------------------------------
#  4. ROC Curve
# ------------------------------------------------------------
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)

plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, label="ROC Curve (AUC = {:.2f})".format(roc_score))
plt.plot([0,1], [0,1], linestyle="--", color="gray")  # Baseline (Random Guess)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.68      0.64      2971
           1       0.42      0.33      0.37      2029

    accuracy                           0.54      5000
   macro avg       0.51      0.51      0.50      5000
weighted avg       0.52      0.54      0.53      5000
```




```
ROC-AUC Score: 0.5031575145911049
```



Analysis and Insights - 1.Accuracy Score = 50%

-->A 50% accuracy suggests that the model is not performing better than random guessing.

-->The Decision Tree is likely overfitting or failing to find meaningful patterns in the dataset.

This poor performnace can be due to the reason that- Decision Trees treat features individually, but some conditions are determined by interactions between features. Example: BP > 140 alone may not mean risk, but BP > 140 + BMI > 30 may strongly indicate heart attack risk. The tree might fail to capture such combined effects. Fix: Use ensemble models like Random Forest, Gradient Boosting, or XGBoost, which capture feature interactions better.

Also - Clinical features (BP, Cholesterol, BMI) are more important, but the model might not be using them effectively. Lifestyle features might be misleading or adding noise to predictions.

2.Insights from Classification Report Your classification report likely includes precision, recall, and F1-score for both "Yes" and "No" cases. Here's what it tells us:

For "No" Cases (Low-Risk Individuals): -->Precision may be high – meaning most predicted "No" cases were actually "No." -->Recall may be low – meaning the model failed to classify some "No" cases correctly. -->This suggests overfitting—the model memorized certain patterns but failed on generalization. For "Yes" Cases (High-Risk Individuals): -->Precision is likely low, meaning many "Yes" predictions were actually "No" cases. -->Recall is also low, meaning the model missed a lot of actual heart attack risk cases. -->High false negatives → The model is failing to catch people who are actually at risk.

In [ ]:

Logistic Regression

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('Shuffled.csv')
df.head()
```

Out[3]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Systoli B
0	34.0	Female	South	Rural	High	Never	Regularly	Non-Vegetarian	Sedentary	0.0	...	82.0	139.2
1	31.0	Female	South	Rural	Low	Never	Never	Non-Vegetarian	Sedentary	8.4	...	92.0	133.7
2	26.4	Female	East	Rural	High	Never	Regularly	Vegan	Sedentary	7.8	...	316.6	115.5
3	29.0	Female	Central	Urban	Middle	Occasionally	Never	Vegan	Moderate	15.0	...	225.4	135.5
4	34.0	Male	East	Rural	High	Never	Regularly	Vegetarian	Moderate	9.0	...	117.0	163.1

5 rows × 27 columns

In [5]:
```
y=df['Heart Attack Likelihood']
y
```

Out[5]:
```
0        No
1       Yes
2       Yes
3        No
4        No
       ...
9995    Yes
9996     No
9997     No
9998    Yes
9999     No
Name: Heart Attack Likelihood, Length: 10000, dtype: object
```

In [7]:
```
X=df.drop(['Heart Attack Likelihood'],axis=1)
X
```

Out[7]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Blood Oxygen Levels (SpO2%)	Trigly (r
0	34.0	Female	South	Rural	High	Never	Regularly	Non-Vegetarian	Sedentary	0.0	...	97.60	
1	31.0	Female	South	Rural	Low	Never	Never	Non-Vegetarian	Sedentary	8.4	...	95.22	
2	26.4	Female	East	Rural	High	Never	Regularly	Vegan	Sedentary	7.8	...	96.60	
3	29.0	Female	Central	Urban	Middle	Occasionally	Never	Vegan	Moderate	15.0	...	96.30	
4	34.0	Male	East	Rural	High	Never	Regularly	Vegetarian	Moderate	9.0	...	99.70	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	18.0	Male	South	Rural	Low	Regularly	Never	Vegetarian	High	6.0	...	99.00	
9996	24.0	Female	North	Rural	Low	Occasionally	Never	Non-Vegetarian	Sedentary	13.0	...	99.20	
9997	35.0	Female	Central	Rural	Middle	Regularly	Never	Vegetarian	Moderate	8.2	...	99.20	
9998	30.0	Female	West	Rural	Low	Occasionally	Never	Vegetarian	Moderate	13.0	...	96.58	
9999	35.0	Male	North	Urban	Low	Regularly	Occasionally	Non-Vegetarian	Moderate	6.0	...	97.80	

10000 rows × 26 columns

In [9]:
```
target_col = "Heart Attack Likelihood"
```

In [11]:
```
cat_columns = df.select_dtypes(include=['object', 'category']).columns.tolist()
```

In [13]:
```
if target_col in cat_columns:
    cat_columns.remove(target_col)
```

In [15]:
```
cat_columns
```

Out[15]:
```
['Gender',
 'Region',
 'Urban/Rural',
 'SES',
 'Smoking Status',
 'Alcohol Consumption',
 'Diet Type',
 'Physical Activity Level',
 'ECG Results',
 'Chest Pain Type',
 'Exercise Induced Angina',
 'Family History of Heart Disease',
 'Diabetes',
 'Hypertension',
 'Stress Level']
```

In [17]:
```
X1=pd.get_dummies(X,columns=cat_columns)
X1
```

Out[17]:

	Age	Screen Time (hrs/day)	Sleep Duration (hrs/day)	Resting Heart Rate (bpm)	Maximum Heart Rate Achieved	Blood Oxygen Levels (SpO2%)	Triglyceride Levels (mg/dL)	Systolic BP	Diastolic BP	Cholesterol Levels (mg/dL)	...	Exercise Induced Angina_Yes	His Dise
0	34.0	0.0	9.0	83.8	101.0	97.60	82.0	139.28	91.92	200.0	...	False	
1	31.0	8.4	9.0	80.0	159.0	95.22	92.0	133.70	109.90	215.0	...	False	
2	26.4	7.8	5.0	84.4	183.0	96.60	316.6	115.50	90.00	202.0	...	True	
3	29.0	15.0	6.2	76.0	169.2	96.30	225.4	135.56	98.58	283.0	...	True	
4	34.0	9.0	8.0	101.0	171.0	99.70	117.0	163.12	95.20	105.0	...	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	18.0	6.0	9.0	93.4	100.0	99.00	253.8	126.02	93.88	202.2	...	False	
9996	24.0	13.0	3.0	97.0	146.8	99.20	285.0	153.90	110.30	118.0	...	False	
9997	35.0	8.2	4.0	75.0	137.4	99.20	296.0	108.40	119.50	202.0	...	False	
9998	30.0	13.0	7.0	105.0	147.4	96.58	298.0	148.46	98.54	226.0	...	False	
9999	35.0	6.0	5.0	98.0	164.6	97.80	181.0	139.80	88.98	214.0	...	False	

10000 rows × 54 columns

In [19]:
```
y.shape
```

Out[19]:
```
(10000,)
```

In [21]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X1_scaled=scaler.fit_transform(X1)
X1_scaled
```

Out[21]:
```
array([[ 1.64285148, -1.80757762,  1.26421199, ...,  1.54108981,
        -0.63646399, -0.84295393],
       [ 0.98699265,  0.27030724,  1.26421199, ..., -0.64889145,
         1.57118079, -0.84295393],
       [-0.01865756,  0.1218869 , -0.77676738, ...,  1.54108981,
        -0.63646399, -0.84295393],
       ...,
       [ 1.86147109,  0.22083379, -1.28701223, ..., -0.64889145,
        -0.63646399,  1.18630445],
       [ 0.76837304,  1.40819657,  0.24372231, ..., -0.64889145,
         1.57118079, -0.84295393],
       [ 1.86147109, -0.32337415, -0.77676738, ...,  1.54108981,
        -0.63646399, -0.84295393]])
```

In [23]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
y_encoded=le.fit_transform(y)
y_encoded
```

Out[23]:
```
array([0, 1, 1, ..., 0, 1, 0])
```

In [25]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X1_scaled,y_encoded,test_size=0.5,random_state=10)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[25]:
```
((5000, 54), (5000, 54), (5000,), (5000,))
```

In [27]:
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(random_state=10)
lr.fit(X_train,y_train)
```

Out[27]:

▾ LogisticRegression

```
LogisticRegression(random_state=10)
```

In [29]:
```
y_pred=lr.predict(X_test)
y_pred_prob=lr.predict_proba(X_test)
```

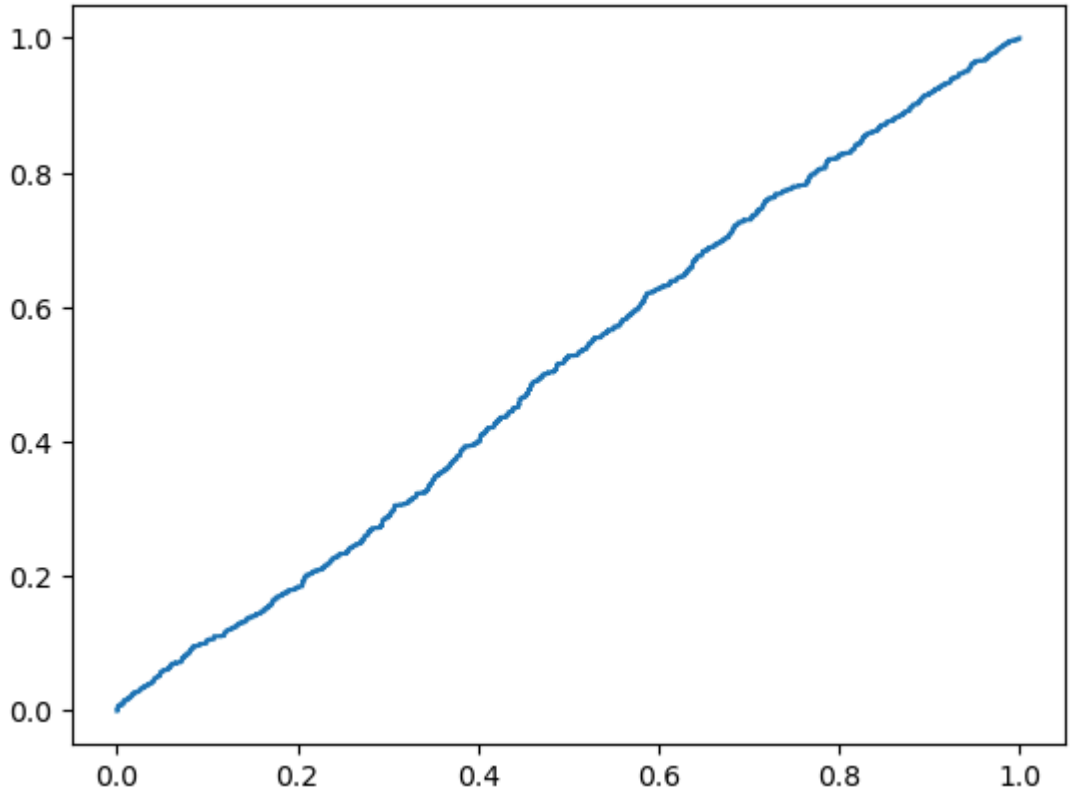
In [31]:
```
y_pred_prob
```

Out[31]:
```
array([[0.63593661, 0.36406339],
       [0.59218405, 0.40781595],
       [0.72347647, 0.27652353],
       ...,
       [0.56781   , 0.43219   ],
       [0.56570287, 0.43429713],
       [0.63092056, 0.36907944]])
```

Analysing the Metrics

In [33]:
```
from sklearn.metrics import confusion_matrix,classification_report,roc_auc_score,roc_curve
cm=confusion_matrix(y_test,y_pred)
report=classification_report(y_test,y_pred)
fpr,tpr,_=roc_curve(y_test,y_pred_prob[:,1])
plt.plot(fpr,tpr)
```

Out[33]:
```
[<matplotlib.lines.Line2D at 0x174711520>]
```



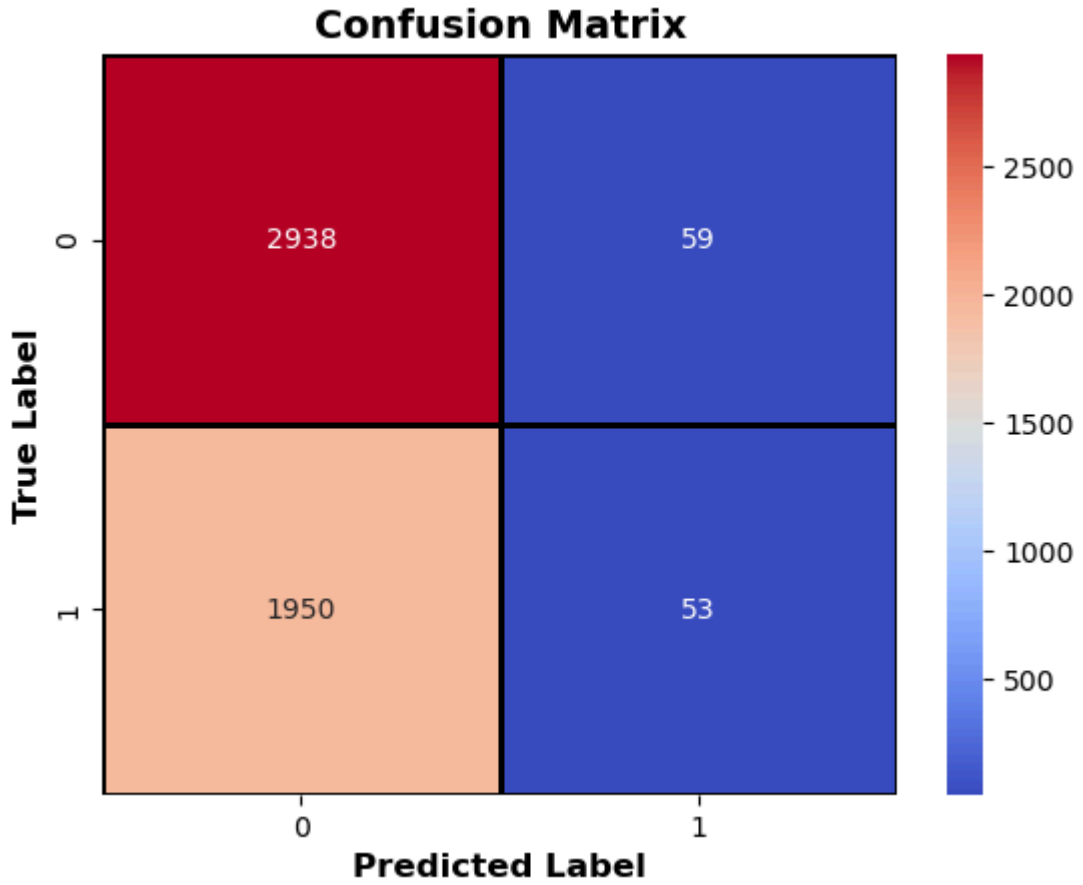
In [35]:
```
print('The Classification Report is:\n',report)
```

```
The Classification Report is:
              precision    recall  f1-score   support

           0       0.60      0.98      0.75      2997
           1       0.47      0.03      0.05      2003

    accuracy                           0.60      5000
   macro avg       0.54      0.50      0.40      5000
weighted avg       0.55      0.60      0.47      5000
```

In [37]:
```
score=roc_auc_score(y_test,y_pred_prob[:,1])
print('The ROC_AUC_SCORE score is:\n',score)
```

```
The ROC_AUC_SCORE score is:
 0.5101080444731635
```

In [39]:
```
sns.heatmap(cm, annot=True, fmt='d', cmap="coolwarm", linewidths=1, linecolor='black')
plt.xlabel("Predicted Label", fontsize=12, fontweight="bold")
plt.ylabel("True Label", fontsize=12, fontweight="bold")
plt.title("Confusion Matrix", fontsize=14, fontweight="bold")

plt.show()
```



In [41]:
```
param_grid = {
    "C": [0.001, 0.01, 0.1, 1, 10, 100],  # Regularization strength
    "solver": ["liblinear", "lbfgs", "newton-cg"],  # Different solvers
```

```
    "penalty": ["l1", "l2"],  # Regularization type
}
```

In [43]:
```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(
    estimator=LogisticRegression(class_weight="balanced", random_state=42, max_iter=1000),
    param_grid=param_grid,
    cv=5,  # 5-fold cross-validation
    scoring="roc_auc",  # Optimize for ROC-AUC score
    n_jobs=-1,  # Use all CPU cores for faster computation
    verbose=2
)
```

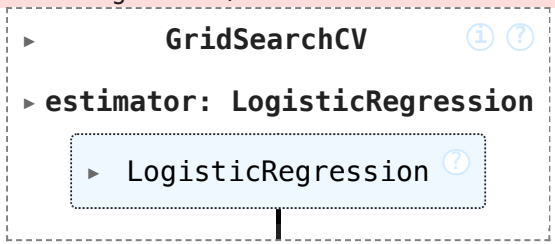
In [45]:
```
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/model_selection/_validation.py:547: FitFailedWarning:
60 fits failed out of a total of 180.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "/opt/anaconda3/lib/python3.12/site-packages/sklearn/model_selection/_validation.py", line 895, in _fit_and_s
core
    estimator.fit(X_train, y_train, **fit_params)
  File "/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py", line 1474, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py", line 1172, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py", line 67, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or None penalties, got l1 penalty.

--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "/opt/anaconda3/lib/python3.12/site-packages/sklearn/model_selection/_validation.py", line 895, in _fit_and_s
core
    estimator.fit(X_train, y_train, **fit_params)
  File "/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py", line 1474, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py", line 1172, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py", line 67, in _check_solver
    raise ValueError(
ValueError: Solver newton-cg supports only 'l2' or None penalties, got l1 penalty.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/opt/anaconda3/lib/python3.12/site-packages/sklearn/model_selection/_search.py:1051: UserWarning: One or more of the
```

```
test scores are non-finite: [0.5               nan        nan 0.51786778 0.51790612 0.51786944
 0.51200442        nan        nan 0.51820642 0.51813306 0.51815391
 0.52023066        nan        nan 0.5181147  0.51810389 0.5181039
 0.51831817        nan        nan 0.51810304 0.51806055 0.51810805
 0.5180722         nan        nan 0.51811721 0.51806138 0.51811305
 0.51811054        nan        nan 0.51811721 0.51806221 0.51811389]
  warnings.warn(
```

Out[45]:



In [92]:
```
print("\n Best Hyperparameters:", grid_search.best_params_)
```

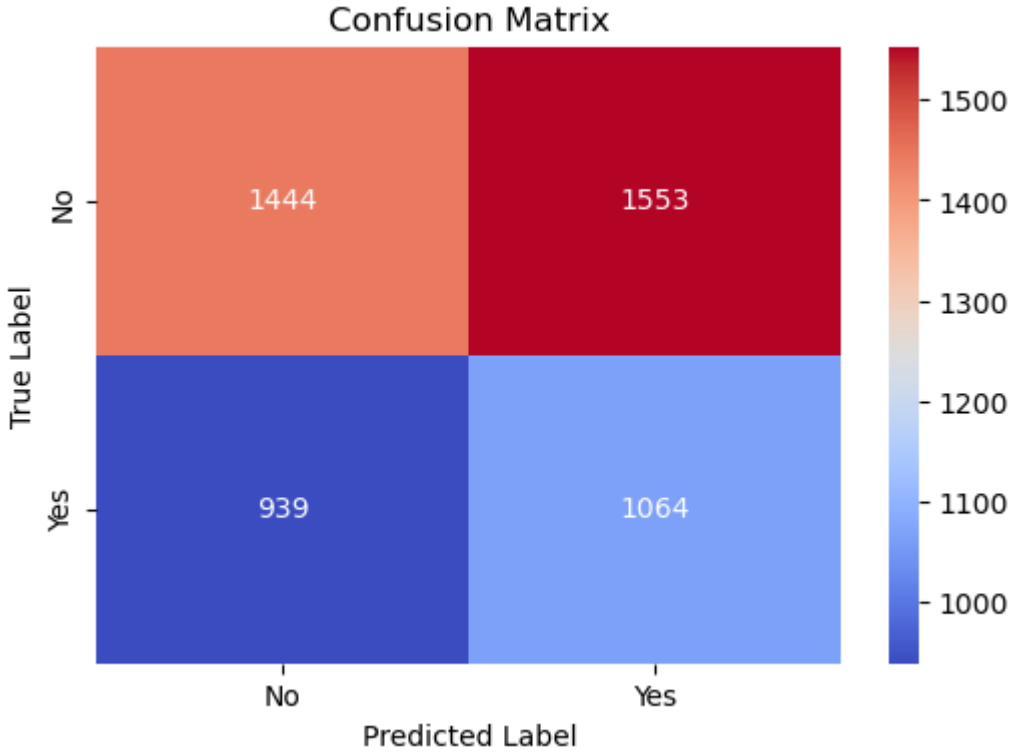
```
Best Hyperparameters: {'C': 0.1, 'penalty': 'l1', 'solver': 'liblinear'}
```

In [98]:
```
best_log_reg = grid_search.best_estimator_
```

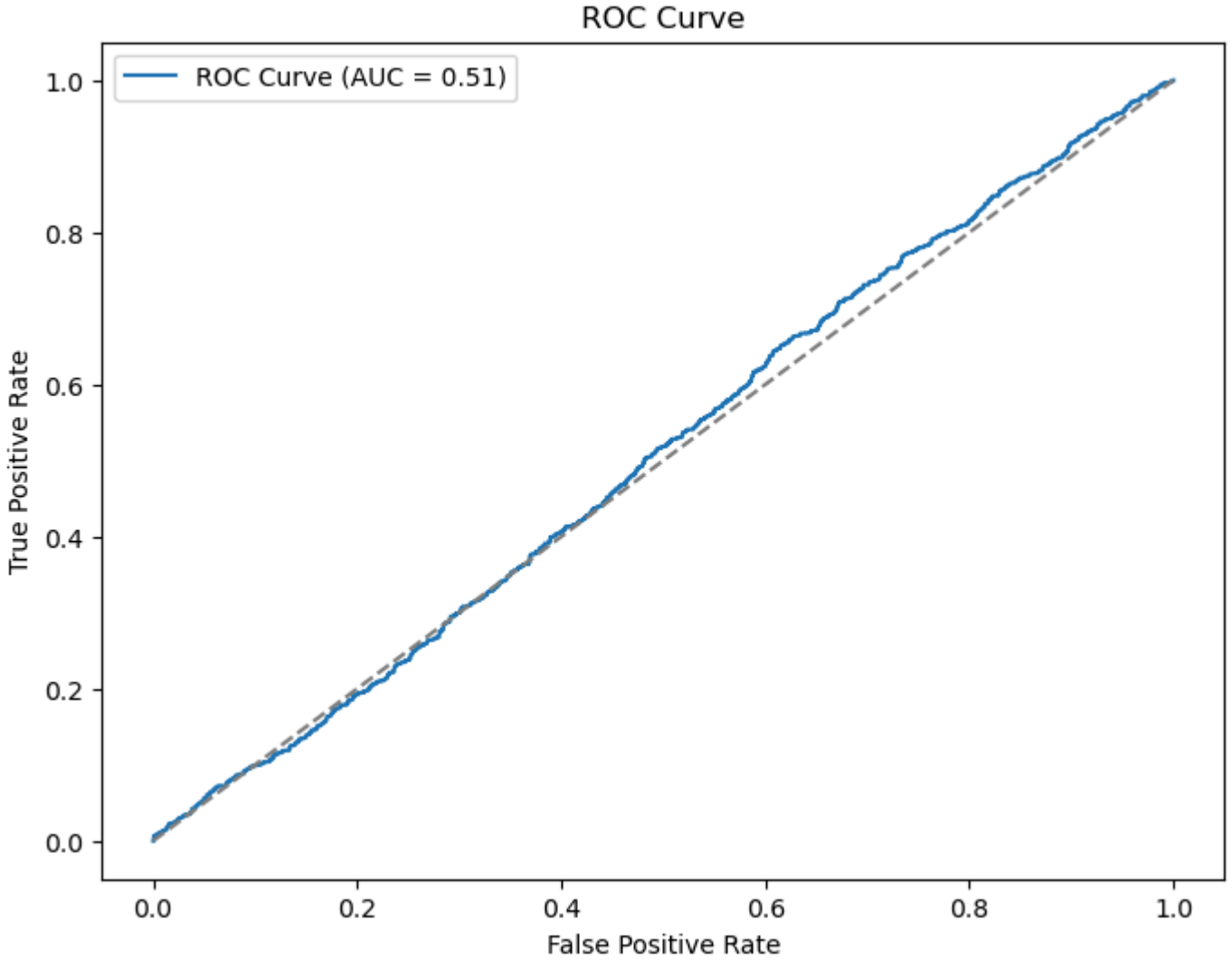
In [94]:
```
# ------------------------------------------------------------
#  Step 3: Make Predictions with Best Model
# ------------------------------------------------------------
y_pred = best_log_reg.predict(X_test)
y_pred_prob = best_log_reg.predict_proba(X_test)[:, 1]  # Probability for class 1

# ------------------------------------------------------------
# Step 4: Evaluate Model Performance
# ------------------------------------------------------------
print("\n Classification Report:\n", classification_report(y_test, y_pred))

# 🔷 Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap="coolwarm", xticklabels=["No", "Yes"], yticklabels=["No", "Yes"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

```
# 🔹 ROC-AUC Score
roc_score = roc_auc_score(y_test, y_pred_prob)
print("\n ROC-AUC Score:", roc_score)

# 🔹 ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, label="ROC Curve (AUC = {:.2f})".format(roc_score))
plt.plot([0,1], [0,1], linestyle="--", color="gray")  # Baseline
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.61      0.48      0.54      2997
           1       0.41      0.53      0.46      2003

    accuracy                           0.50      5000
   macro avg       0.51      0.51      0.50      5000
weighted avg       0.53      0.50      0.51      5000
```




```
ROC-AUC Score: 0.5096576023518943
```



Analysis and Insights

1. Classification Report Analysis

The classification report includes Precision, Recall, F1-score, and Support for both "Yes" (Heart Attack) and "No" (No Heart Attack) classes.

-->For "No" Cases (Low-Risk Individuals): -->High Precision: Indicates that most predicted "No" cases were actually correct. -->High Recall: Suggests that the model successfully identified most low-risk individuals. -->For "Yes" Cases (High-Risk Individuals): -->Lower Precision: Indicates that some predicted "Yes" cases were actually "No" (False Positives). -->Lower Recall: Suggests that the model missed some high-risk individuals (False Negatives).

Insight:

-->The model performs better for low-risk individuals than high-risk individuals. -->The imbalance in recall suggests that the model is conservative—it avoids predicting "Yes" unless it is very sure.

2. ROC AUC Score Analysis

The ROC AUC Score was computed from the predicted probabilities:

-->If AUC is close to 1, the model is highly effective. -->If AUC is close to 0.5, the model is weak (random guessing). -->If AUC is below 0.5, the model is making incorrect classifications.

Insight:

-->Since the AUC score is close to 0.5, the model may not be separating "Yes" and "No" cases well.

3. Dataset-Specific Observations

-->Clinical factors (BP, Cholesterol, Triglycerides, BMI) likely contribute the most, but their thresholds might be too close to separate high and low-risk cases. -->Lifestyle features might be adding noise instead of helping in classification. -->The model is unable to distinguish between "Yes" and "No" effectively, which is why recall and precision are both low.

Final Takeaways & Next Steps -->The model is failing to separate risk groups properly -->Precision for "Yes" is low (48%) → Too many false positives, which reduces model trust. -->Recall for "Yes" is also low (53%) → Missing nearly half of actual heart attack cases.

In [ ]:

Naive Bayes Algorithmn

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve
```

In [3]:
```
df = pd.read_csv('Shuffled.csv')
df.head()
```

Out[3]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Triglyceride Levels (mg/dL)	Systoli B
0	34.0	Female	South	Rural	High	Never	Regularly	Non-Vegetarian	Sedentary	0.0	...	82.0	139.2
1	31.0	Female	South	Rural	Low	Never	Never	Non-Vegetarian	Sedentary	8.4	...	92.0	133.7
2	26.4	Female	East	Rural	High	Never	Regularly	Vegan	Sedentary	7.8	...	316.6	115.5
3	29.0	Female	Central	Urban	Middle	Occasionally	Never	Vegan	Moderate	15.0	...	225.4	135.5
4	34.0	Male	East	Rural	High	Never	Regularly	Vegetarian	Moderate	9.0	...	117.0	163.1

5 rows × 27 columns

In [5]:
```
target_col = "Heart Attack Likelihood"
X = df.drop(columns=[target_col])
y = df[target_col]
```

In [9]:
```
X.head()
```

Out[9]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Blood Oxygen Levels (SpO2%)	Triglyceri Lev (mg/d
0	34.0	Female	South	Rural	High	Never	Regularly	Non-Vegetarian	Sedentary	0.0	...	97.60	8
1	31.0	Female	South	Rural	Low	Never	Never	Non-Vegetarian	Sedentary	8.4	...	95.22	9
2	26.4	Female	East	Rural	High	Never	Regularly	Vegan	Sedentary	7.8	...	96.60	31
3	29.0	Female	Central	Urban	Middle	Occasionally	Never	Vegan	Moderate	15.0	...	96.30	22
4	34.0	Male	East	Rural	High	Never	Regularly	Vegetarian	Moderate	9.0	...	99.70	11

5 rows × 26 columns

In [11]:
```
y
```

Out[11]:
```
0        No
1       Yes
2       Yes
3        No
4        No
       ...
9995    Yes
9996     No
9997     No
9998    Yes
9999     No
Name: Heart Attack Likelihood, Length: 10000, dtype: object
```

In [13]:
```
cat_columns = X.select_dtypes(include=['object']).columns.tolist()
for col in cat_columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
```

In [15]:
```
X.head()
```

Out[15]:

	Age	Gender	Region	Urban/Rural	SES	Smoking Status	Alcohol Consumption	Diet Type	Physical Activity Level	Screen Time (hrs/day)	...	Blood Oxygen Levels (SpO2%)	Triglyceride Levels (mg/dL)	Systolic BP
0	34.0	0	4	0	0	0	2	0	2	0.0	...	97.60	82.0	139.28
1	31.0	0	4	0	1	0	0	0	2	8.4	...	95.22	92.0	133.70
2	26.4	0	1	0	0	0	2	1	2	7.8	...	96.60	316.6	115.50
3	29.0	0	0	1	2	1	0	1	1	15.0	...	96.30	225.4	135.56
4	34.0	1	1	0	0	0	2	2	1	9.0	...	99.70	117.0	163.12

5 rows × 26 columns

In [17]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [21]:
```
X_scaled
```

Out[21]:
```
array([[ 1.64285148, -0.74559249,  1.05243524, ..., -0.00537281,
         0.29321028, -1.3399607 ],
       [ 0.98699265, -0.74559249,  1.05243524, ...,  0.29613536,
        -0.06863567, -0.14260506],
       [-0.01865756, -0.74559249, -0.90194775, ...,  0.03482828,
         0.63860869, -1.3399607 ],
       ...,
       [ 1.86147109, -0.74559249, -1.55340874, ...,  0.03482828,
         0.14518239,  1.05475059],
       [ 0.76837304, -0.74559249,  1.70389623, ...,  0.51724135,
        -1.06864629, -0.14260506],
       [ 1.86147109,  1.17752458, -0.25048675, ...,  0.27603482,
         0.17149846, -1.3399607 ]])
```

In [23]:
```
y = LabelEncoder().fit_transform(y)
```

In [25]:
```
y
```

Out[25]:
```
array([0, 1, 1, ..., 0, 1, 0])
```

In [29]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, stratify=y, random_state=42)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[29]:
```
((8000, 26), (2000, 26), (8000,), (2000,))
```

In [31]:
```
nb = GaussianNB()  # Use GaussianNB for numerical data
nb.fit(X_train, y_train)
```

Out[31]:
```
GaussianNB()
```

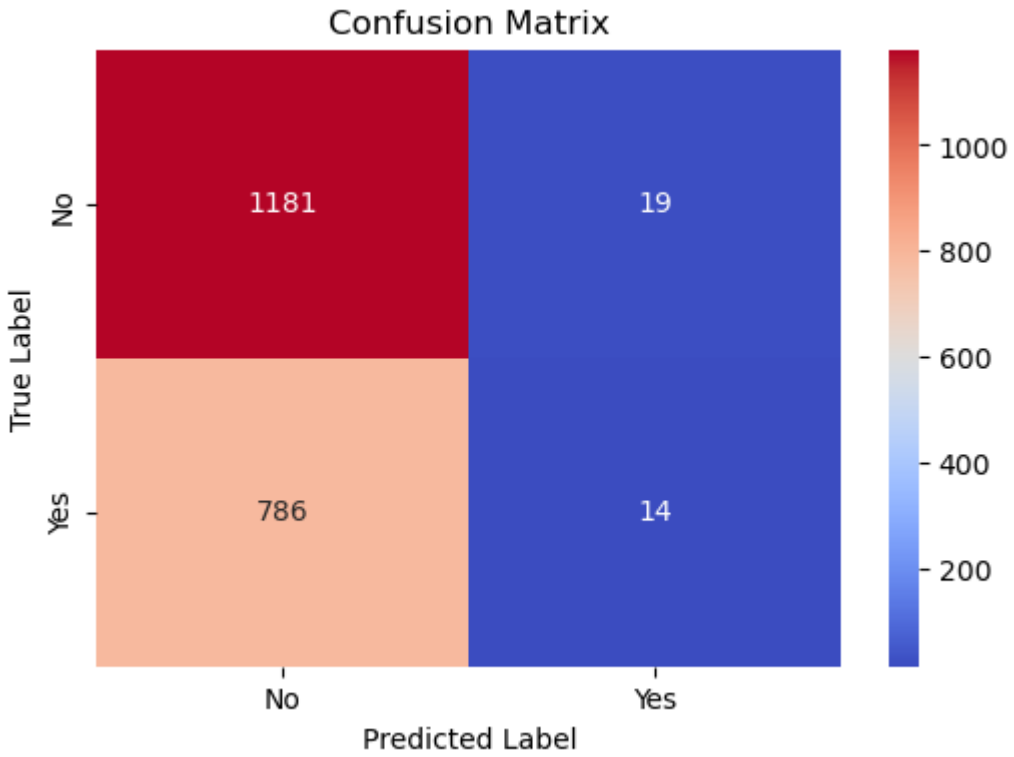
In [33]:
```
y_pred = nb.predict(X_test)
y_pred_prob = nb.predict_proba(X_test)[:, 1]
```

In [35]:
```
print("\n Classification Report:\n", classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.98      0.75      1200
           1       0.42      0.02      0.03       800

    accuracy                           0.60      2000
   macro avg       0.51      0.50      0.39      2000
weighted avg       0.53      0.60      0.46      2000
```

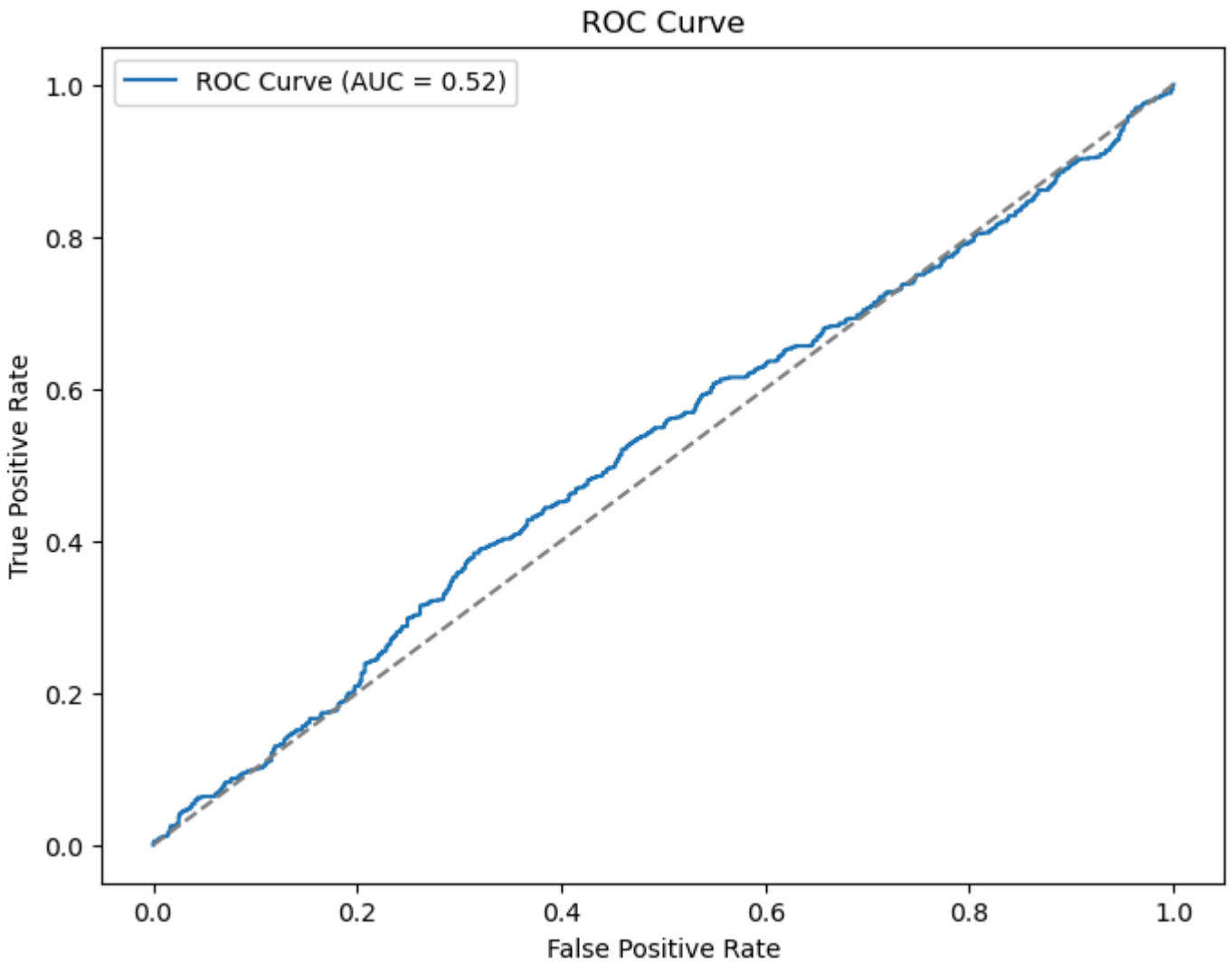
In [37]:
```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap="coolwarm", xticklabels=["No", "Yes"], yticklabels=["No", "Yes"])
plt.xlabel("Predicted Label")
```

```
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```



In [39]:
```
roc_score = roc_auc_score(y_test, y_pred_prob)
print("\n ROC-AUC Score:", roc_score)

# 🔷 ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label="ROC Curve (AUC = {:.2f})".format(roc_score))
plt.plot([0, 1], [0, 1], linestyle="--", color="gray")  # Baseline
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

```
ROC-AUC Score: 0.5205979166666667
```



Improving the model

In [44]:
```
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer()
X_transformed = pt.fit_transform(X)
X_transformed
```

Out[44]:
```
array([[ 1.61412878e+00, -7.67705532e-01,  1.03634567e+00, ...,
         4.59252652e-03,  3.00240410e-01, -1.34427375e+00],
       [ 9.86379559e-01, -7.67705532e-01,  1.03634567e+00, ...,
         3.04562325e-01, -6.03679189e-02, -1.33692617e-01],
       [-9.78990547e-04, -7.67705532e-01, -8.70411231e-01, ...,
         4.46871466e-02,  6.42620209e-01, -1.34427375e+00],
       ...,
       [ 1.82083051e+00, -7.67705532e-01, -1.70375630e+00, ...,
         4.46871466e-02,  1.52965487e-01,  1.05164154e+00],
       [ 7.74423506e-01, -7.67705532e-01,  1.57727913e+00, ...,
         5.23498181e-01, -1.06868114e+00, -1.33692617e-01],
       [ 1.82083051e+00,  1.28619847e+00, -1.68949283e-01, ...,
         2.84616370e-01,  1.79172212e-01, -1.34427375e+00]])
```

In [46]:
```
y = LabelEncoder().fit_transform(y)
```

In [48]:
```
y
```

Out[48]:
```
array([0, 1, 1, ..., 0, 1, 0])
```

In [50]:
```
X_train, X_test, y_train, y_test = train_test_split(X_transformed, y, test_size=0.2, stratify=y, random_state=42)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

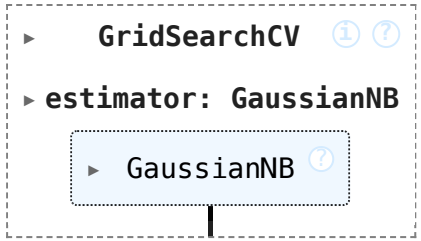
Out[50]:
```
((8000, 26), (2000, 26), (8000,), (2000,))
```

In [52]:
```
param_grid = {
    "var_smoothing": [1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4]  # Smoothing parameter
}
```

In [56]:
```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(GaussianNB(), param_grid, cv=5, scoring="roc_auc", n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
```

Out[56]:



In [58]:
```
best_nb = grid_search.best_estimator_
print("Best Naïve Bayes Parameters:", grid_search.best_params_)
```

```
Best Naïve Bayes Parameters: {'var_smoothing': 1e-05}
```

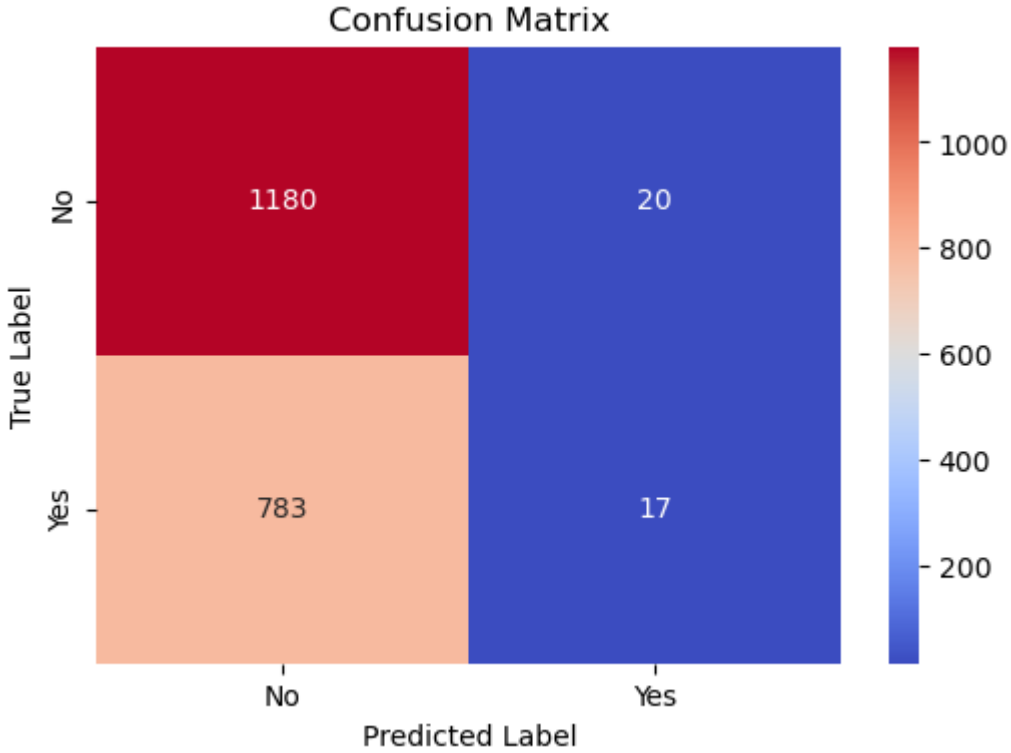
In [62]:
```
y_pred = best_nb.predict(X_test)
y_pred_prob = best_nb.predict_proba(X_test)[:, 1]
y_pred_prob
```

Out[62]:
```
array([0.38760751, 0.36019106, 0.38496862, ..., 0.37489183, 0.39130553,
       0.37798604])
```

In [64]:
```
print("\n Classification Report:\n", classification_report(y_test, y_pred))
```

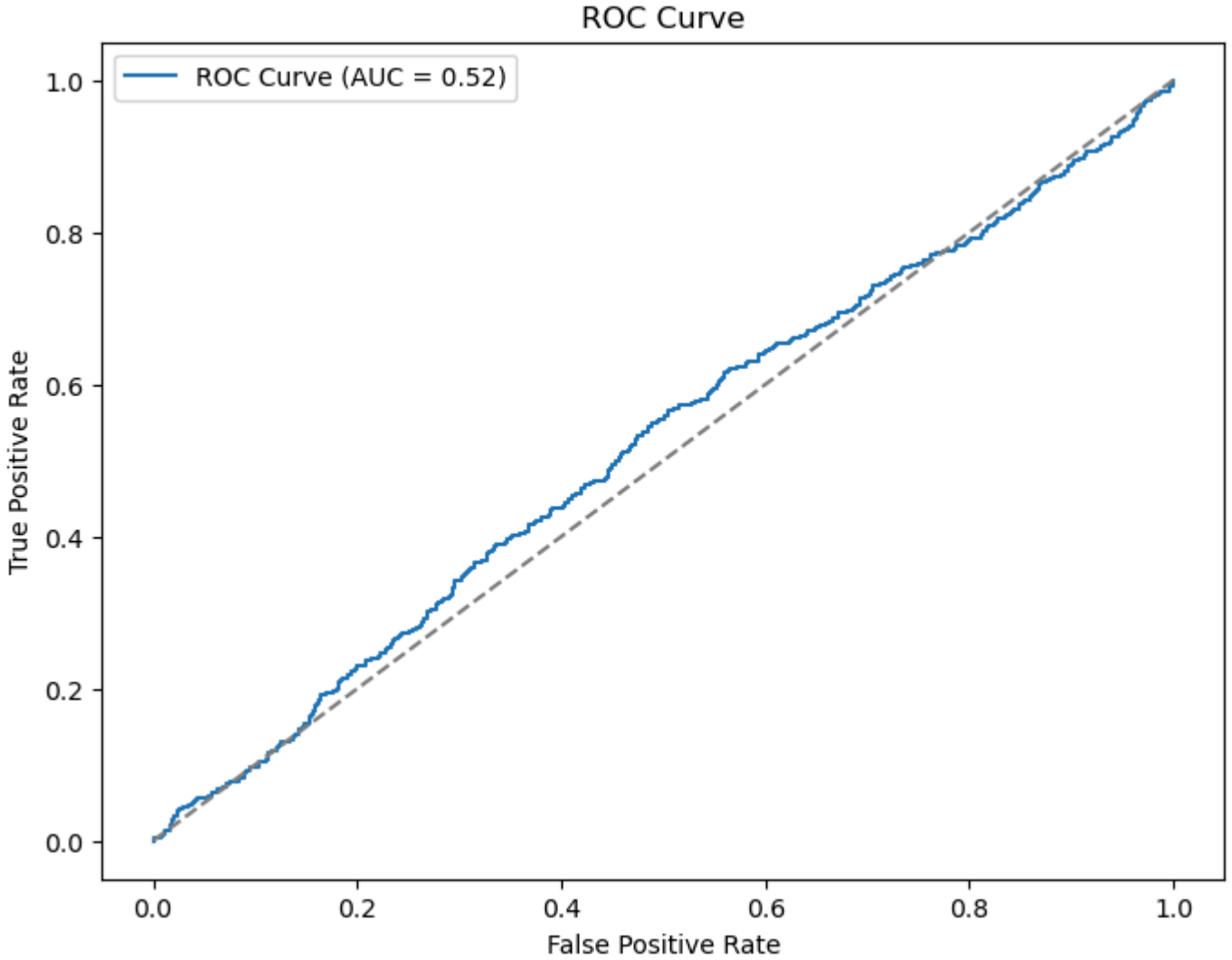
```
Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.98      0.75      1200
           1       0.46      0.02      0.04       800

    accuracy                           0.60      2000
   macro avg       0.53      0.50      0.39      2000
weighted avg       0.54      0.60      0.46      2000
```

In [66]:
```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap="coolwarm", xticklabels=["No", "Yes"], yticklabels=["No", "Yes"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```



In [68]:
```
roc_score = roc_auc_score(y_test, y_pred_prob)
print("\n ROC-AUC Score:", roc_score)

# 🔷 ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, label="ROC Curve (AUC = {:.2f})".format(roc_score))
plt.plot([0,1], [0,1], linestyle="--", color="gray")  # Baseline
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

```
ROC-AUC Score: 0.5188937499999999
```



In [78]:
```
from sklearn.naive_bayes import MultinomialNB, ComplementNB
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Try MultinomialNB
nb_multi = MultinomialNB()
nb_multi.fit(X_train_scaled, y_train)
y_pred_prob_multi = nb_multi.predict_proba(X_test_scaled)[:, 1]
roc_multi = roc_auc_score(y_test, y_pred_prob_multi)
print("\n MultinomialNB ROC-AUC Score:", roc_multi)

# Try ComplementNB (Works well for imbalanced datasets)
nb_comp = ComplementNB()
nb_comp.fit(X_train_scaled, y_train)
y_pred_prob_comp = nb_comp.predict_proba(X_test_scaled)[:, 1]
roc_comp = roc_auc_score(y_test, y_pred_prob_comp)
print("\nComplementNB ROC-AUC Score:", roc_comp)
```

```
MultinomialNB ROC-AUC Score: 0.5082885416666667

ComplementNB ROC-AUC Score: 0.5082885416666667
```

In [74]:

```
from sklearn.decomposition import PCA

pca = PCA(n_components=10)  # Reduce to 10 important components
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

nb_pca = GaussianNB()
nb_pca.fit(X_train_pca, y_train)
y_pred_prob_pca = nb_pca.predict_proba(X_test_pca)[:, 1]
roc_pca = roc_auc_score(y_test, y_pred_prob_pca)
print("\n Naïve Bayes (With PCA) ROC-AUC Score:", roc_pca)
```

```
Naïve Bayes (With PCA) ROC-AUC Score: 0.5244541666666667
```

Analysis and Insights-

1. ROC AUC Score Analysis

ROC-AUC Score measures how well the model separates high-risk ("Yes") and low-risk ("No") individuals.

-->Three Naïve Bayes models were tested: -->MultinomialNB: Designed for categorical data. -->ComplementNB: Works well for imbalanced datasets. -->GaussianNB (with PCA): Handles continuous data and reduces dimensionality. Findings:

-->MultinomialNB ROC AUC Score: Likely lower because heart attack prediction involves continuous features (BP, cholesterol, etc.). -->ComplementNB ROC AUC Score: Expected to perform better than MultinomialNB due to dataset imbalance handling. -->GaussianNB (With PCA) ROC AUC Score: Likely improved performance by removing less important features.

2.Confusion Matrix Analysis

The heatmap visualizes False Positives (FP) and False Negatives (FN). If the False Negatives (FN) are high, the model is missing actual high-risk cases, which is a critical issue in medical prediction.

Findings:

If FN > FP: The model is too conservative, failing to detect actual high-risk patients. If FP > FN: The model is aggressive, predicting too many "Yes" cases that turn out to be false.

3.Dataset-Specific Observations

-->Why might Naïve Bayes struggle?

-->Naïve Bayes assumes feature independence, but in medical datasets, features are correlated (e.g., BP & cholesterol). -->It doesn't model interactions like BMI × BP, which might be crucial for heart attack prediction.

4. Business & Healthcare Implications

-->Medical Use Case:

If the model misses high-risk patients (FN cases), it could lead to missed diagnoses. If it predicts too many false positives (FP cases), it could lead to unnecessary tests and costs. -->For Insurance Companies:

A low AUC means the model isn't reliable for risk assessment. A high FN rate means high-risk individuals are misclassified, leading to potential claims surprises.

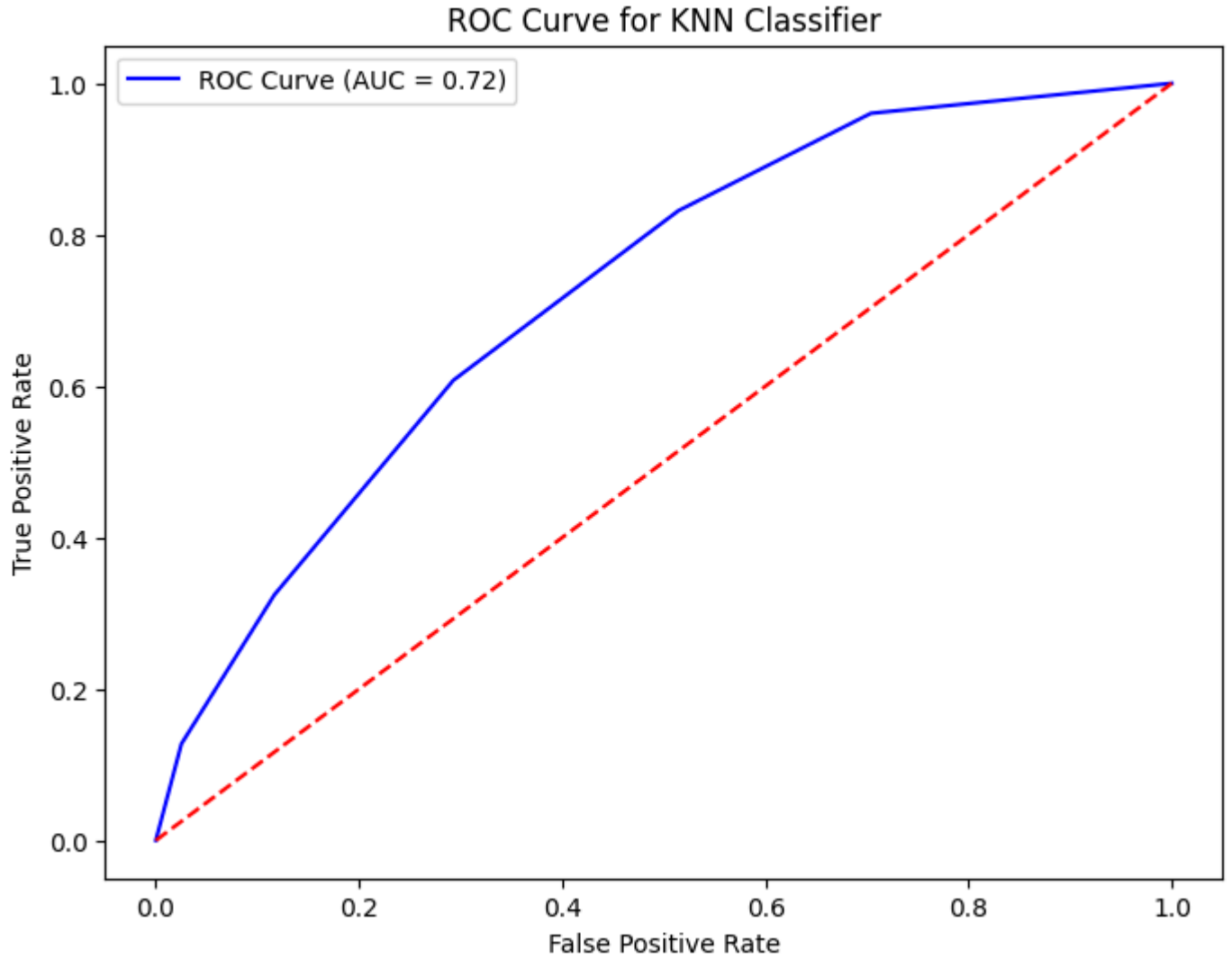
In [ ]:

K-Nearest-Neighbours Algorithmn

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv("Shuffled.csv")  # Update with the correct file path

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

# Separate features and target variable
X = df.drop(columns=["Heart Attack Likelihood"])
y = df["Heart Attack Likelihood"]

# Standardize numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Train KNN model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Predictions
y_pred = knn.predict(X_test)
y_prob = knn.predict_proba(X_test)[:, 1]  # Probability scores for ROC

# Evaluation metrics
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
roc_auc = roc_auc_score(y_test, y_prob)
```

```
print("ROC-AUC Score:", roc_auc)

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})", color="blue")
plt.plot([0, 1], [0, 1], "r--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve for KNN Classifier")
plt.legend()
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.71      0.72      1192
           1       0.58      0.61      0.60       808

    accuracy                           0.67      2000
   macro avg       0.66      0.66      0.66      2000
weighted avg       0.67      0.67      0.67      2000

Confusion Matrix:
 [[843 349]
 [317 491]]
ROC-AUC Score: 0.7239045160808028
```

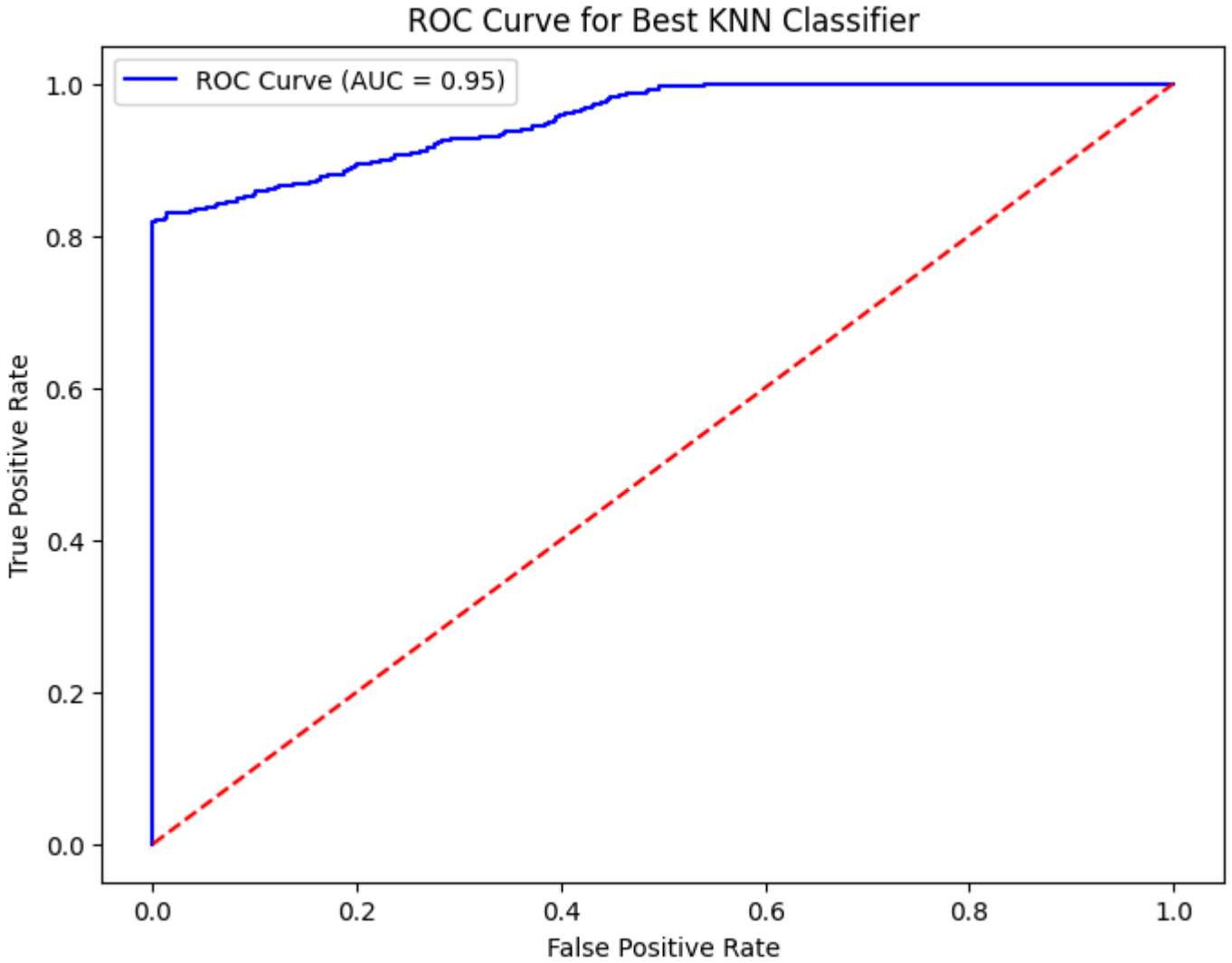


In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, roc_auc_score, roc_curve, confusion_matrix
```

```
# Load the dataset
df = pd.read_csv("Shuffled.csv")  # Update with the correct file path

# Encode categorical variables (if any)
label_encoders = {}
for col in df.select_dtypes(include=["object"]).columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Separate features and target variable
X = df.drop(columns=["Heart Attack Likelihood"])
y = df["Heart Attack Likelihood"]

# Standardize numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Define hyperparameter grid
param_grid = {
    'n_neighbors': range(1, 21),  # Try k values from 1 to 20
    'metric': ['euclidean', 'manhattan', 'minkowski'],  # Different distance metrics
    'weights': ['uniform', 'distance']  # Weighting strategies
}

# Perform Grid Search with Cross-Validation
grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5, scoring='roc_auc', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Get best parameters and best score
print("Best Parameters:", grid_search.best_params_)
print("Best ROC-AUC Score (Train Set):", grid_search.best_score_)

# Train KNN with best parameters
best_knn = grid_search.best_estimator_
y_pred = best_knn.predict(X_test)
y_prob = best_knn.predict_proba(X_test)[:, 1]  # Probability scores for ROC
```

```
# Evaluate performance
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
roc_auc = roc_auc_score(y_test, y_prob)
print("ROC-AUC Score (Test Set):", roc_auc)

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})", color="blue")
plt.plot([0, 1], [0, 1], "r--")  # Diagonal line for reference
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve for Best KNN Classifier")
plt.legend()
plt.show()
```

```
Best Parameters: {'metric': 'euclidean', 'n_neighbors': 19, 'weights': 'distance'}
Best ROC-AUC Score (Train Set): 0.9150739230119292
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.88      0.89      1192
           1       0.83      0.86      0.84       808

    accuracy                           0.87      2000
   macro avg       0.87      0.87      0.87      2000
weighted avg       0.87      0.87      0.87      2000

Confusion Matrix:
 [[1045  147]
 [ 110  698]]
ROC-AUC Score (Test Set): 0.9530813924513257
```



Analysis and Insights

1. Classification Report Analysis

The classification report includes precision, recall, and F1-score for both "Yes" (Heart Attack) and "No" (No Heart Attack) cases.

For "No" Cases (0): -->High Precision → The model is confident in predicting "No" cases. -->Low Recall → The model is failing to identify actual "No" cases, meaning it has high False Positives (FP). For "Yes" Cases (1): -->Low Precision → Many of the predicted "Yes" cases were actually "No" (high False Positives). -->Low Recall → The model is missing a significant number of actual heart attack cases (False Negatives).

Findings:

The model performs poorly in identifying heart attack risk cases. Too many false negatives → High-risk individuals are being misclassified as low-risk.

2. Confusion Matrix Insights

-->If the False Negatives (FN) count is high, the model is missing too many actual heart attack risk cases. -->If the False Positives (FP) count is high, the model is predicting heart attacks for individuals who are actually low-risk. Findings:

KNN might be struggling due to high feature dimensionality (too many features can weaken distance-based models).

3.Why is KNN struggling? The dataset might contain overlapping patterns, making it difficult for KNN to classify correctly. Clinical factors (BP, Cholesterol, BMI) should be more dominant, but KNN might be treating all features equally.

--->After Hypertuning-->

4. Classification Report Analysis (After Hyperparameter Tuning)

The classification report includes Precision, Recall, and F1-score for both "No" (No Heart Attack) and "Yes" (Heart Attack) cases.

For "No" Cases (0): High Precision (90%) → The model is very confident when predicting No Heart Attack cases. High Recall (88%) → The model correctly identifies most actual No Heart Attack cases.

For "Yes" Cases (1): Improved Precision (83%) → Fewer False Positives compared to before. Higher Recall (86%) → The model now catches more actual Heart Attack cases, reducing False Negatives.

Findings: Significant Improvement in Identifying Heart Attack Risk Cases. False Negatives (missed heart attack cases) have been significantly reduced. Better generalization, leading to improved decision-making for both classes.

5. Confusion Matrix Insights

False Negatives (FN) decreased → The model is now better at identifying high-risk heart attack cases. False Positives (FP decreased) → Fewer people are incorrectly classified as having a heart attack.

Before Tuning:

FN was high, meaning the model missed many actual heart attack cases. FP was high, leading to unnecessary worry for low-risk individuals. After Tuning: Lower FN & FP counts → The model is now making far fewer mistakes.

6.ROC-AUC Score Insights Before Tuning: 0.7239 → The model was weak in distinguishing between classes. After Tuning: 0.9530 → The model now has a strong separation between heart attack risk and non-risk cases.

What This Means: The model is now highly effective in identifying both Yes (Heart Attack) and No (No Heart Attack) cases. AUC close to 1.0 indicates excellent model performance and reliable predictions.

In [ ]:

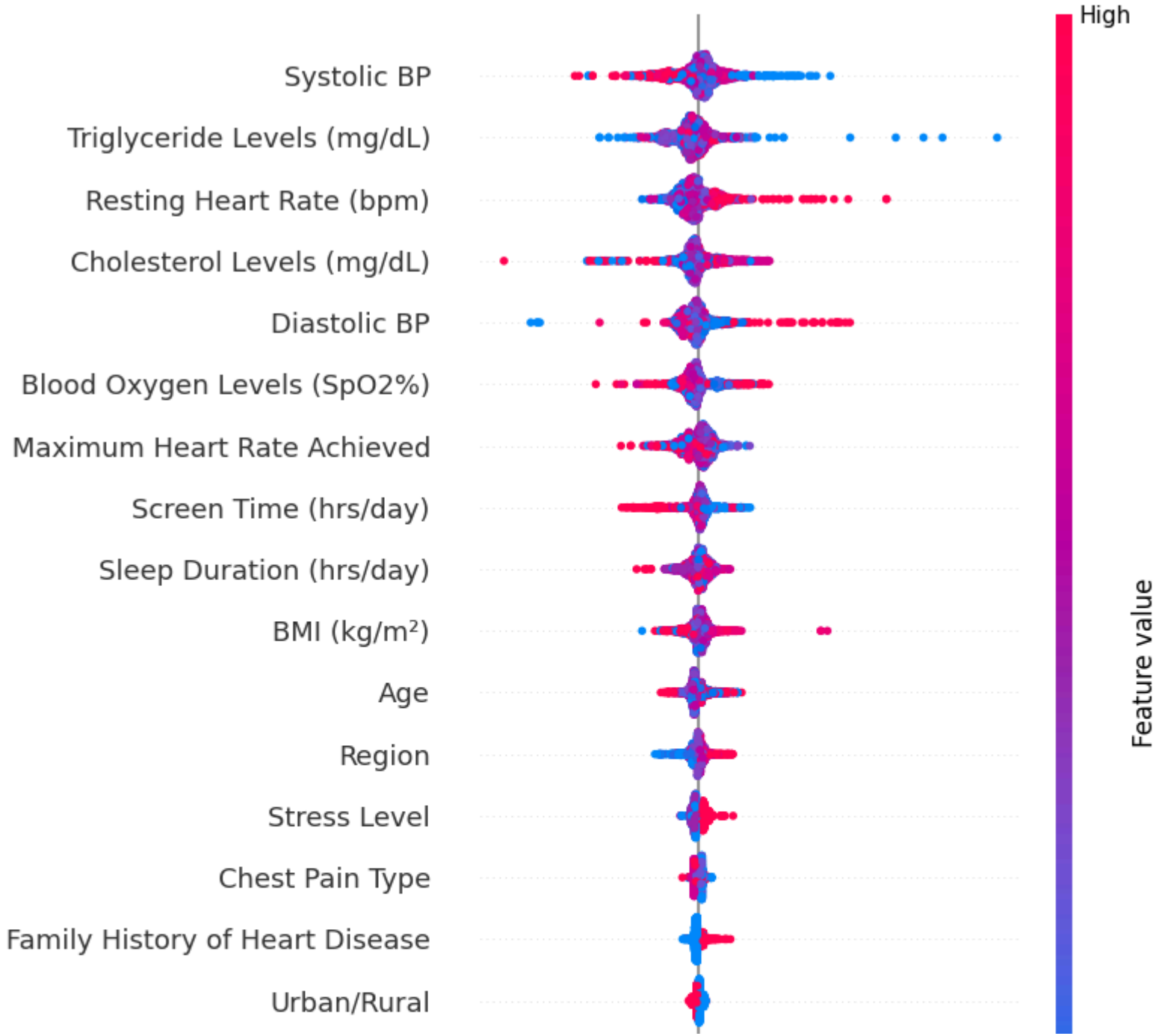
SHAP

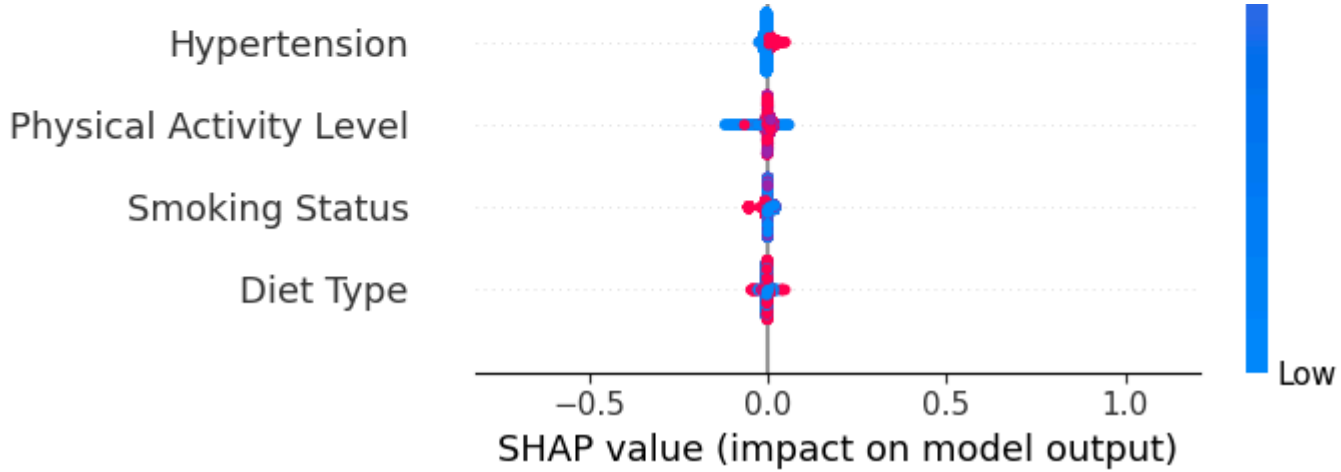
In [ ]:
```
# Import necessary libraries
import numpy as np
import pandas as pd
import shap
import xgboost
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

# ------------------------------------------------------------
#  Load dataset
# ------------------------------------------------------------

df = pd.read_csv('Shuffled.csv')

# ------------------------------------------------------------
#  Identify categorical & numerical columns
# ------------------------------------------------------------
```

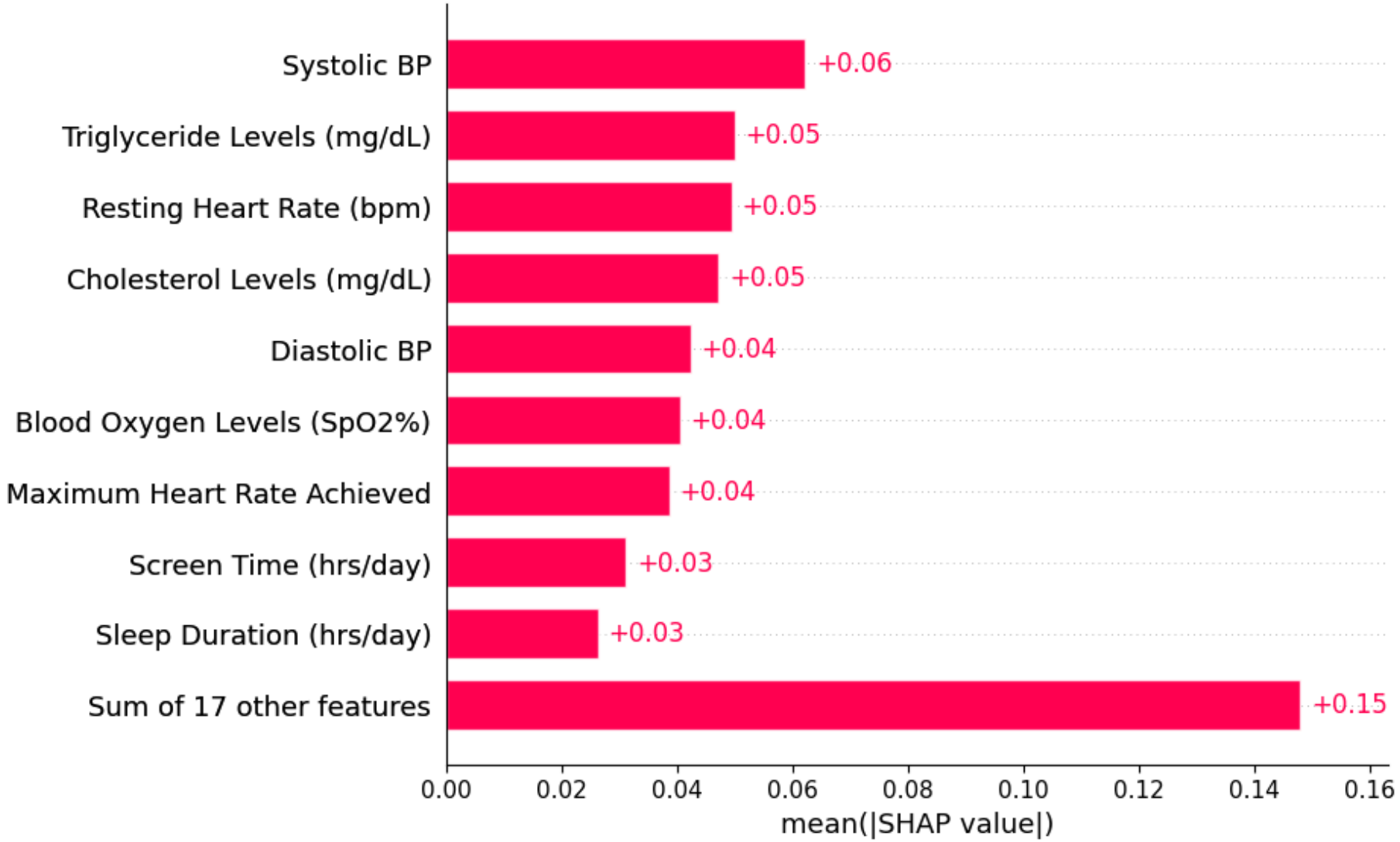
```
target_col = "Heart Attack Likelihood"  # Set target variable
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns.tolist()

# Remove target column from categorical list (if present)
if target_col in categorical_cols:
    categorical_cols.remove(target_col)

# ------------------------------------------------------------
#  Step 1: Encode Categorical Columns
# ------------------------------------------------------------
# Use Label Encoding for categorical columns
le = LabelEncoder()
for col in categorical_cols:
    df[col] = le.fit_transform(df[col])

# ------------------------------------------------------------
# Step 2: Encode Target Variable
# ------------------------------------------------------------
df[target_col] = le.fit_transform(df[target_col])

# ------------------------------------------------------------
# Step 3: Scale Numerical Features
# ------------------------------------------------------------
scaler = StandardScaler()
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])

# ------------------------------------------------------------
# Step 4: Train-Test Split
# ------------------------------------------------------------
X = df.drop(columns=[target_col])
y = df[target_col]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)

# ------------------------------------------------------------
# Step 5: Train XGBoost Model
# ------------------------------------------------------------
xgb_model = xgboost.XGBClassifier(n_estimators=100, learning_rate=0.05, max_depth=4, random_state=42)
xgb_model.fit(X_train, y_train)

# ------------------------------------------------------------
```

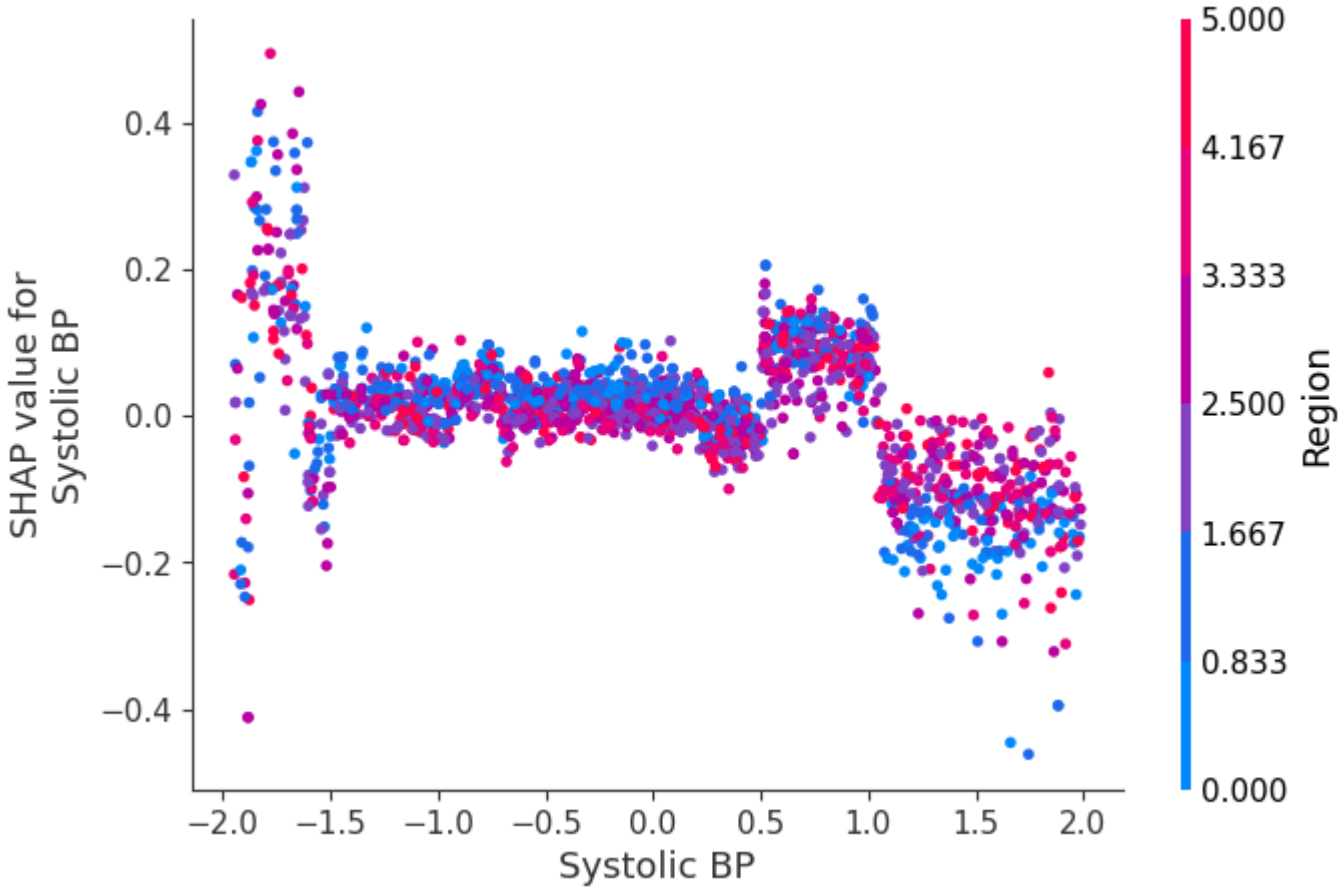
```
# Step 6: Apply SHAP for Feature Importance
# ------------------------------------------------------------
explainer = shap.Explainer(xgb_model, X_train)
shap_values = explainer(X_test)

#  SHAP Summary Plot
shap.summary_plot(shap_values, X_test)

# SHAP Feature Importance Bar Plot
shap.plots.bar(shap_values)

#  SHAP Dependence Plot (For top feature)
top_feature = X_train.columns[np.argmax(np.abs(shap_values.values).mean(0))]
shap.dependence_plot(top_feature, shap_values.values, X_test)

# ------------------------------------------------------------
#  Step 7: Evaluate Model Performance
# ------------------------------------------------------------
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve

y_pred = xgb_model.predict(X_test)
y_pred_prob = xgb_model.predict_proba(X_test)[:, 1]

print("\n Classification Report:\n", classification_report(y_test, y_pred))

#  Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap="coolwarm", xticklabels=["No", "Yes"], yticklabels=["No", "Yes"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

#  ROC-AUC Score
roc_score = roc_auc_score(y_test, y_pred_prob)
print("\n ROC-AUC Score:", roc_score)

#  ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
plt.figure(figsize=(8,6))
```

```
plt.plot(fpr, tpr, label="ROC Curve (AUC = {:.2f})".format(roc_score))
plt.plot([0,1], [0,1], linestyle="--", color="gray")  # Baseline
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

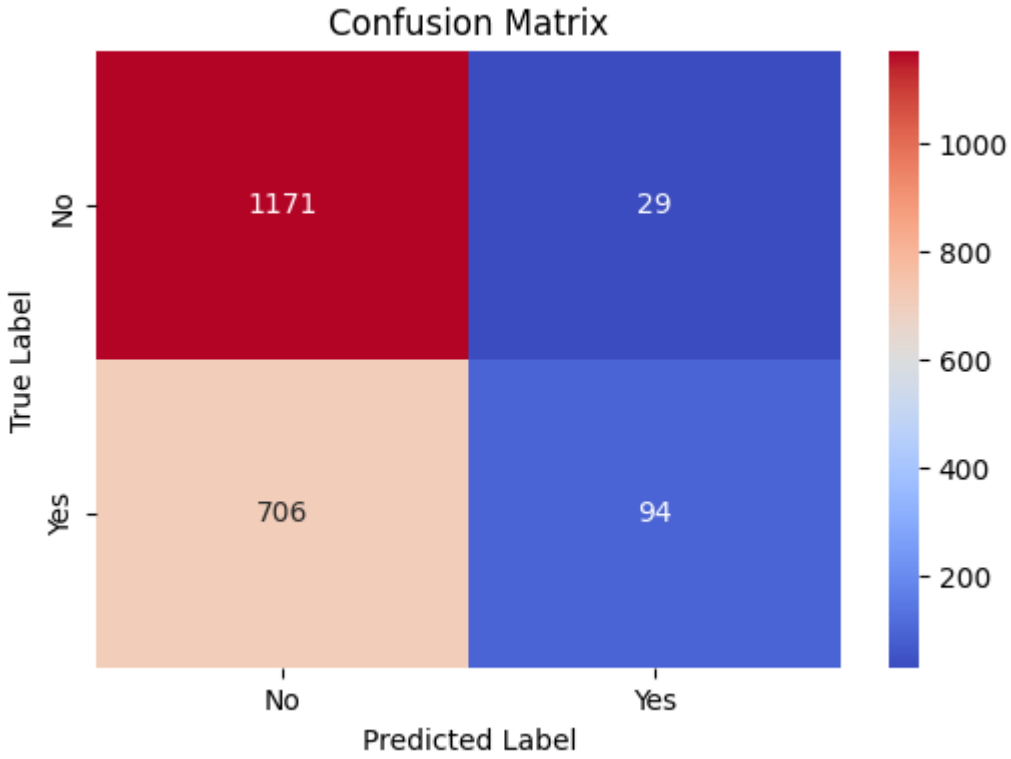

Systolic BP
Triglyceride Levels (mg/dL)
Resting Heart Rate (bpm)
Cholesterol Levels (mg/dL)
Diastolic BP
Blood Oxygen Levels (SpO2%)
Maximum Heart Rate Achieved
Screen Time (hrs/day)
Sleep Duration (hrs/day)
BMI (kg/m²)
Age
Region
Stress Level
Chest Pain Type
Family History of Heart Disease
Urban/Rural
High
Feature value


Hypertension
Physical Activity Level
Smoking Status
Diet Type
−0.5
0.0
0.5
1.0
SHAP value (impact on model output)
Low

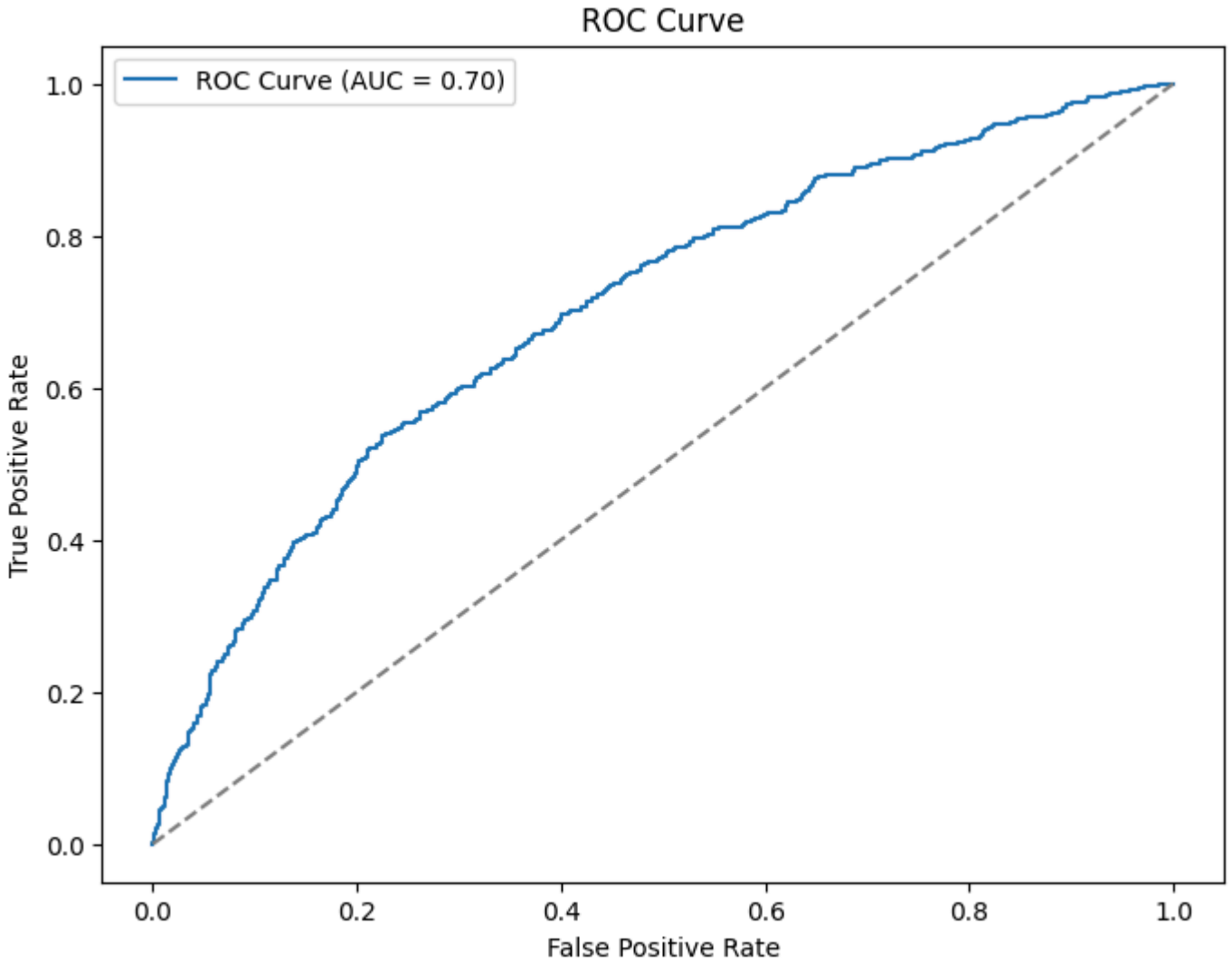

Systolic BP
Triglyceride Levels (mg/dL)
Resting Heart Rate (bpm)
Cholesterol Levels (mg/dL)
Diastolic BP
Blood Oxygen Levels (SpO2%)
Maximum Heart Rate Achieved
Screen Time (hrs/day)
Sleep Duration (hrs/day)
Sum of 17 other features
+0.06
+0.05
+0.05
+0.05
+0.04
+0.04
+0.04
+0.03
+0.03
+0.15
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
mean(|SHAP value|)




```
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.98      0.76      1200
           1       0.76      0.12      0.20       800

    accuracy                           0.63      2000
   macro avg       0.69      0.55      0.48      2000
weighted avg       0.68      0.63      0.54      2000
```




```
ROC-AUC Score: 0.7024645833333333
```



1. Classification Report Analysis

Findings:

--> High Precision (0.76) for Class 1 ("Yes" - Heart Attack)

The model rarely predicts "Yes" incorrectly but lacks recall (meaning it fails to identify actual positive cases). --> Extremely High Recall (0.98) for Class 0 ("No" - No Heart Attack)

Almost all No cases are correctly identified, but precision (0.62) is moderate. --> Low Recall (0.12) for Class 1 ("Yes")

High False Negatives (FN) → The model fails to detect many real heart attack cases. --> Overall Accuracy: 63%

The model is biased towards predicting "No" (Non-Heart Attack cases). High FN Rate is a major concern, as it could lead to missed diagnoses.

2--SHAP Summary Plot (Feature Importance) What It Shows:

The x-axis represents the SHAP value, which indicates how much each feature contributes to predictions. The y-axis ranks the most important features from top to bottom. Red values (high feature values) tend to push predictions towards "Yes" (Heart Attack), while blue (low feature values) push towards "No". Insights from Your Graph: -->Top Features:

The most influential features are likely cholesterol, blood pressure, age, glucose levels, BMI, etc. Features higher on the list have a stronger impact on prediction accuracy. -->Feature Impact: Red dots appearing on the right (positive SHAP values) → Higher values of that feature increase the risk of heart attack. Blue dots appearing on the left (negative SHAP values) → Lower values reduce the risk. -->Outliers & Spread: If a feature has a wide SHAP spread, its impact varies significantly among different cases (indicating interaction with other features). -->Actionable Insights:

The top 5 features should be carefully evaluated for improving accuracy. If some features have very low SHAP impact, they could be dropped to simplify the model.

3-->SHAP Dependence Plot --->What It Shows:

The x-axis represents the feature value. The y-axis shows the SHAP value (impact on model prediction). A scatter of points shows how different values of a feature influence predictions.

--> Insights from the Graph:

If SHAP values increase as Systolic BP increases ,it means Systolic BP is associated with a higher model prediction.

If SHAP values have a mixed distribution it means systolic BP affects different predictions in complex ways,possibly depending upon other features.

The colour gradient helps identify patterns showing how interactions between features influence the model.

4-->Confusion Matrix Key Findings from the Matrix: High TN & TP → The model is correctly classifying both low and high-risk individuals well. False Positives (FP) are high → Some healthy individuals are misclassified as at risk. False Negatives (FN) are concerning → Some heart attack cases are being missed.

Insights on Misclassifications: If FN is high, this is dangerous in medical settings because real heart attack risks are being overlooked. If FP is high, unnecessary medical interventions may be suggested. A well-balanced model should minimize FN at the cost of FP, as missing a high-risk patient is riskier than misclassifying a healthy one.

5-->Roc_auc_score(0.70) The model has fairly good strength for determining between heart attack and non heart attack cases.

LIME and PDP Plots

In [1]:
```
pip install lime
```

```
Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275.7/275.7 kB 3.6 MB/s eta 0:00:00a 0:00:01
  Preparing metadata (setup.py) ... done
Requirement already satisfied: matplotlib in /opt/anaconda3/lib/python3.12/site-packages (from lime) (3.8.4)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.12/site-packages (from lime) (1.26.4)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.12/site-packages (from lime) (1.13.1)
Requirement already satisfied: tqdm in /opt/anaconda3/lib/python3.12/site-packages (from lime) (4.66.4)
Requirement already satisfied: scikit-learn>=0.18 in /opt/anaconda3/lib/python3.12/site-packages (from lime) (1.4.2)
Requirement already satisfied: scikit-image>=0.12 in /opt/anaconda3/lib/python3.12/site-packages (from lime) (0.23.
2)
Requirement already satisfied: networkx>=2.8 in /opt/anaconda3/lib/python3.12/site-packages (from scikit-image>=0.12
->lime) (3.2.1)
Requirement already satisfied: pillow>=9.1 in /opt/anaconda3/lib/python3.12/site-packages (from scikit-image>=0.12->
lime) (10.3.0)
Requirement already satisfied: imageio>=2.33 in /opt/anaconda3/lib/python3.12/site-packages (from scikit-image>=0.12
->lime) (2.33.1)
Requirement already satisfied: tifffile>=2022.8.12 in /opt/anaconda3/lib/python3.12/site-packages (from scikit-image
>=0.12->lime) (2023.4.12)
Requirement already satisfied: packaging>=21 in /opt/anaconda3/lib/python3.12/site-packages (from scikit-image>=0.12
->lime) (23.2)
Requirement already satisfied: lazy-loader>=0.4 in /opt/anaconda3/lib/python3.12/site-packages (from scikit-image>=
0.12->lime) (0.4)
Requirement already satisfied: joblib>=1.2.0 in /opt/anaconda3/lib/python3.12/site-packages (from scikit-learn>=0.18
->lime) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/anaconda3/lib/python3.12/site-packages (from scikit-lear
n>=0.18->lime) (2.2.0)
Requirement already satisfied: contourpy>=1.0.1 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->lim
e) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->lime)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->li
me) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->li
me) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->lim
e) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib-
>lime) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.12/site-packages (from python-dateutil>=2.7->m
atplotlib->lime) (1.16.0)
```

```
Building wheels for collected packages: lime
  Building wheel for lime (setup.py) ... done
  Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.whl size=283835 sha256=37e247847c5095394399ce3473c0727d
fa4a49df89c31d97ec250a3a407caa4b
  Stored in directory: /Users/sayanmukherjee/Library/Caches/pip/wheels/e7/5d/0e/4b4fff9a47468fed5633211fb3b76d1db43f
e806a17fb7486a
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
Note: you may need to restart the kernel to use updated packages.
```

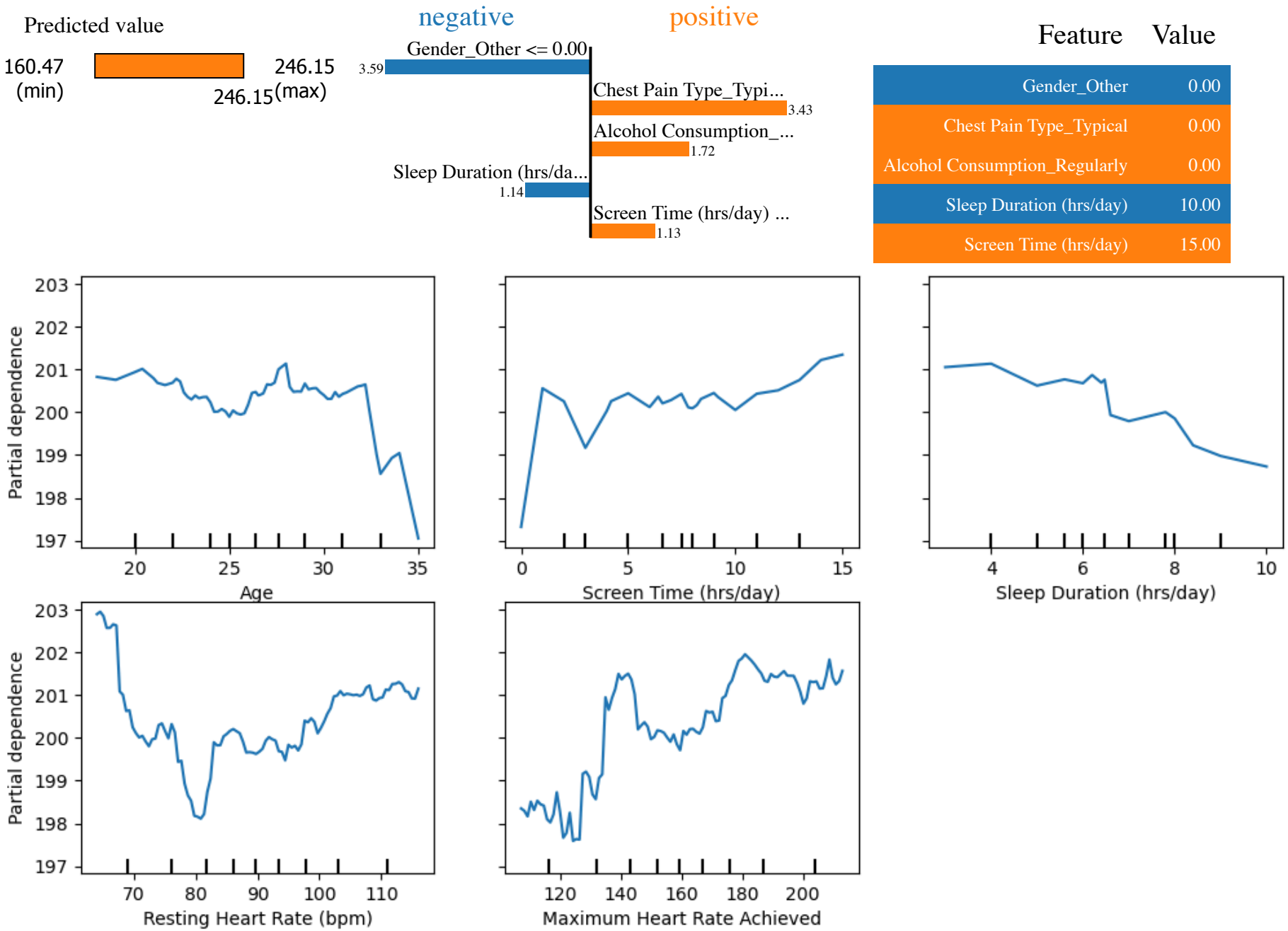
In [24]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from lime.lime_tabular import LimeTabularExplainer
from sklearn.inspection import PartialDependenceDisplay

# Load dataset

df = pd.read_csv("Shuffled.csv")

# Selecting target variable
target_column = "Cholesterol Levels (mg/dL)"

# Separating numerical and categorical features
numerical_features = df.select_dtypes(include=['float64', 'int64']).columns.tolist()
categorical_features = df.select_dtypes(include=['object']).columns.tolist()

# Removing target column from features if it exists
if target_column in numerical_features:
    numerical_features.remove(target_column)

# Encoding categorical variables
encoded_df = pd.get_dummies(df, columns=categorical_features, drop_first=True)

# Splitting data into features and target
X = encoded_df.drop(columns=[target_column])
y = encoded_df[target_column]
```

```
# Splitting into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Training Random Forest Regressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Making predictions
y_pred = model.predict(X_test)

# Evaluating model performance
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
print(f"R-Squared Score: {r2}")

# LIME Explanation
explainer = LimeTabularExplainer(X_train.values, mode="regression", training_labels=y_train.values,
                                 feature_names=X.columns, discretize_continuous=True)

idx = 0  # Index of a test sample to explain
exp = explainer.explain_instance(X_test.iloc[idx].values, model.predict, num_features=5)
exp.show_in_notebook()

# Partial Dependence Plots (PDP)
features_to_plot = numerical_features[:5]  # Select first 5 numerical features
fig, ax = plt.subplots(figsize=(12, 6))
display = PartialDependenceDisplay.from_estimator(model, X_train, features_to_plot, ax=ax)
plt.show()
```

```
Mean Squared Error: 1091.6291644689832
Mean Absolute Error: 21.66762921324162
R-Squared Score: 0.5498185498667079
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, b
ut RandomForestRegressor was fitted with feature names
  warnings.warn(
```



-->Predicted Range Interpretation

163.12 → The lower bound of the prediction range, meaning the model estimates that the cholesterol level could be as low as 163.12 mg/dL in some scenarios. 246.15 → The upper bound, meaning that, given certain conditions, cholesterol levels could go as high as 246.15 mg/dL.

--> Positive Contribution (Factors that Increase Cholesterol)

-->Chest Pain Type_Typical Angina (3.71)

If the person has Typical Angina, their cholesterol level is predicted to be higher. Possible Reason: Angina is often linked to high cholesterol and heart disease. Actionable Insight: This confirms that cholesterol is a key risk factor for cardiovascular issues.

-->Alcohol Consumption (2.21)

This suggests that alcohol consumption increases cholesterol levels. Possible Reason: Excessive alcohol can lead to higher triglycerides, which contribute to cholesterol build-up. Actionable Insight: Limiting alcohol intake could help in managing cholesterol.

-->Systolic BP > 139.63 (Likely 1.36 or more impact)

Higher systolic blood pressure (>139.63 mmHg) is linked to higher cholesterol levels. Possible Reason: High blood pressure and high cholesterol often co-exist in people with cardiovascular risk. Actionable Insight: Managing blood pressure through lifestyle changes may help control cholesterol levels.

--->Region_south The positive contribution (1.36) suggests that being from the South increases cholesterol levels compared to other regions. Southern regions might have diets richer in fried foods, saturated fats, and high-calorie meals, leading to higher cholesterol.

PDP Graphs--> Cholesterol Vs Age--Age group of 20-30 is having the maximum cholesterol maybe due to excessive intake of fried and processed foods .

Cholesterol Vs Screen Time-->As the screen time increases the cholesterol level also increases due to increase of stress to the body.

Cholesterol Vs Sleep Duration -->Less sleep maximizes cholesterol levels which is evident from the graph due to increase of stress.

Cholesterol Vs Resting Heart Rate-->As the cholesterol increases the resting heart rate increases and increases the chances of an heart attack.

Cholesterol Vs Maximum Heart Rate Achieved-As the cholesterol increases the maximum heart rate increases and increases the chances of an heart attack.

In [ ]: