Assessing Best Fit Model and factors that Contribute to Prediction Of Heart Attack Risk in Global Populations

Aaron Yang Modupeola Fagbenro
Bharath Genji Mohana Ranga
Kundana Chowdary Cherukuri

Abstract :This study seeks to create and assess a predictive model for heart attack risk utilizing the "Heart Attack Prediction Dataset" sourced from Kaggle. The dataset encompasses 8,763 patient records, incorporating variables such as age, sex, cholesterol levels, and lifestyle factors. Through the application of statistical and machine learning techniques, including Logistic Regression, Random Forest, and Gradient Boosting Classifier, the research conducts an Exploratory Data Analysis (EDA) to unveil the critical determinants influencing heart attack risk.

Table of contents

Introduction

The integration of data-driven methodologies in healthcare has opened up novel pathways for disease management and prediction. Leveraging historical medical records, predictive modeling exhibits promising capabilities in identifying individuals at heightened risk of specific ailments, like heart attacks. Through the implementation of targeted and timely preventive measures, these models empower healthcare providers to take proactive actions, thereby enhancing patient outcomes.

Background of the Study

Cardiovascular diseases (CVDs) present a major public health challenge as they are one of the world's leading causes of mortality. Heart attacks, also known as myocardial infarctions in medical terminology, are among the most serious of these and frequently cause an abrupt and unanticipated decline in health. The prediction and prevention of heart attacks are important areas of research in modern medicine due to the complexity of factors that can lead to one, from genetic predispositions to lifestyle choices.

This study makes use of a large dataset called the "Heart Attack Prediction Dataset," which was obtained from Kaggle. The dataset includes a variety of variables, including age, sex, blood pressure, cholesterol, and other important health indicators, that are deemed significant in relation to heart attack risk. Our goal is to create a predictive model that can identify people who are more likely to have a heart attack by using sophisticated statistical and machine learning techniques on this dataset.

In addition to discussing the insights gained from the data and assessing the predictive model's performance, this paper also describes the methodology used to analyze the dataset. It also addresses the possible ethical issues and difficulties in applying such predictive models in clinical settings, as well as the implications of these findings in the larger framework of preventive healthcare.

Research Aim and Objectives

This Research aims to understand and combat the global challenge of heart attacks by assessing best fit model to predict heart attack risk .The "Heart Attack Risk Prediction Dataset" is an excellent resource for study and analysis in healthcare and medical data science. The dataset's main goal is to find the best model to predict the risk of heart attacks in individuals based on a variety of health-related characteristics.

Research Smart Questions

1. Which factors most correlate with heart attack risk across various groups, and what patterns contribute to this risk?
2. How do cholesterol levels and obesity relate, and do they impact heart attack risk?
3. How do age and gender influence heart attack risk, and are there specific patterns related to age or gender?
4. Which model best predicts of heart attack risk?

Dataset Introduction:

We collected data from Kaggle.com, and chose the dataset about heart attack prediction. The link is https://www.kaggle.com/datasets/iamsouravbanerjee/hea attack-prediction-dataset/data. This set of data, which includes 8763 information from patients worldwide, comes to an end in a substantial binary classification component that indicates the existence or failure of a heart attack risk, giving a complete resource for predictive analysis and cardiovascular health research.

Code File Introduction:

This python code file includes the below parts to support our research. Part 1: Load and Inspect the Dataset Part 2: Data Cleaning and preprocessing Part 3: Exploratory Data Analysis(EDA) Part 4: Feature Selection Part 5: Modeling Selection and Training Part 6: Model Evaluation (This part is included in the part 5) Part 7: Conclusion, Recommendation, Futurework

Data Columns Selection:

We selected some columns from the original dataset, because there are some unrelated columns for our topic. We deleted the all unrelated columns before we imported data into python.

Data Source: This study utilizes the "Heart Attack Risk Prediction Dataset" obtained from Kaggle.com. The dataset, which can be accessed at Heart Attack Prediction Dataset by Sourav Banerjee, consists of 8,763 entries, each described by the following 16 columns:

Patient ID: A unique identifier for each patient. Age: The age of the patient. Sex: Gender of the patient, categorized as Male or Female. Cholesterol: The cholesterol levels of the patient. Heart Rate: The heart rate of the patient. Diabetes: Indicates whether the patient has diabetes (0: No, 1: Yes). Smoking: Smoking status of the patient (0: Non-smoker, 1: Smoker). Obesity: Obesity status of the patient (0: Not obese, 1: Obese). Alcohol Consumption: Level of alcohol consumption by the patient. Exercise Hours Per Week: The number of hours the patient exercises each week. Diet: Dietary habits of the patient. Medication Use: Medication usage by the patient (0: No, 1: Yes). Stress Level: Stress level reported by the patient. Sedentary Hours Per Day: Daily hours of sedentary activity. Physical Activity Days Per Week: Number of days per week the patient engages in physical activity. Sleep Hours Per Day: Average hours of sleep per day. Heart Attack Risk: Indicates the presence of heart attack risk (0: No, 1: Yes).

The data analysis focuses on cleaning data, preprocessing data, exploring data, and analyzing the results of those graphs.

Literature Review

The recent studies in cardiovascular health have brought forth significant insights into the understanding and management of cardiovascular disease (CVD), particularly focusing on populations with no prior history of CVD, the prediction of lifetime risks based on factors present at middle age, and the identification of key factors associated with CVD through machine learning techniques.

In the first study, researchers aimed to assess the risk of cardiovascular outcomes in elderly populations without prior CVD history. This was achieved by analyzing Medicare data, focusing on comorbidities, lifestyle factors, and healthcare history. The study employed machine learning algorithms, which demonstrated superior discriminatory power compared to traditional models. This finding underscores the importance of integrated care management in elderly populations. This study is detailed in "Cardiovascular disease outcomes and associated risk factors in a Medicare population without prior CVD history"(Reference[1]).

The second study, "Prediction of Lifetime Risk for Cardiovascular Disease by Risk Factor Burden at 50 Years of Age" (Reference [2]), investigated the lifetime risk of CVD based on risk factors present at the age of 50. The research utilized data from the Framingham Heart Study and found that the absence of risk factors at age 50 was linked to a significantly lower lifetime risk of CVD. This highlights the importance of managing risk factors from middle age to reduce the likelihood of developing CVD later in life.

Lastly, the third study focused on identifying key factors associated with CVD in the Kashgar region. This extensive study analyzed over two million adults using logistic regression and machine learning techniques. Key factors identified included age, occupation, hypertension, exercise frequency, and dietary pattern. These findings, documented in "Machine learning identifies prominent factors associated with cardiovascular disease: findings from two million adults in the Kashgar Prospective Cohort Study (KPCS)" (Reference [3]), emphasize the potential of machine learning in identifying and managing CVD risk factors.

Collectively, these studies underscore the importance of early prevention, integrated healthcare, and the innovative use of machine learning in the fight against CVD. They provide valuable insights into how healthcare professionals and researchers can better understand, predict, and manage cardiovascular health risks.

Exploratory Data Analysis

Data Analysis Methodology: To analyze this dataset, many python3 library was employed. The following code was used to read and initially explore the dataset:

Loading necessary libraries and importing the dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

```python
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
import os
```

Load and Inspect the Dataset

```python
# Importing dataset from CSV
df = pd.read_csv('heart_attack_prediction_dataset_revised.csv', delim
df
```

	Patient ID	Age	Sex	Cholesterol	Heart Rate	Diabetes	Smoking	Obe
0	BMW7812	67	Male	208	72	0	1	0
1	CZE1114	21	Male	389	98	1	1	1
2	BNI9906	21	Female	324	72	1	0	0
3	JLN3497	84	Male	383	73	1	1	0
4	GFO8847	66	Male	318	93	1	1	1
…	…	…	…	…	…	…	…	…
8758	MSV9918	60	Male	121	61	1	1	0
8759	QSV6764	28	Female	120	73	1	0	1
8760	XKA5925	47	Male	250	105	0	1	1
8761	EPE6801	36	Male	178	60	1	1	0
8762	ZWN9666	25	Female	356	75	1	0	0

Displaying the first few rows of the dataset

```python
print(df.head())
```

```
  Patient ID  Age     Sex  Cholesterol  Heart Rate  Diabetes  Smoking
0    BMW7812   67    Male          208          72         0        1
1    CZE1114   21    Male          389          98         1        1
2    BNI9906   21  Female          324          72         1        0
3    JLN3497   84    Male          383          73         1        1
4    GFO8847   66    Male          318          93         1        1

   Obesity  Alcohol Consumption  Exercise Hours Per Week       Diet  \
0        0                    0                 4.168189    Average
1        1                    1                 1.813242  Unhealthy
2        0                    0                 2.078353    Healthy
```

```
3       0                    1                 9.828130    Average
4       1                    0                 5.804299  Unhealthy

   Medication Use  Stress Level  Sedentary Hours Per Day  \
0               0             9                 6.615001
1               0             1                 4.963459
2               1             9                 9.463426
3               0             9                 7.648981
4               0             6                 1.514821

   Physical Activity Days Per Week  Sleep Hours Per Day  Heart Attack R
0                                0                    6
1                                1                    7
2                                4                    4
3                                3                    4
4                                1                    5
```

Displaying dataset information

```python
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8763 entries, 0 to 8762
Data columns (total 17 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Patient ID                       8763 non-null   object
 1   Age                              8763 non-null   int64
 2   Sex                              8763 non-null   object
 3   Cholesterol                      8763 non-null   int64
 4   Heart Rate                       8763 non-null   int64
 5   Diabetes                         8763 non-null   int64
 6   Smoking                          8763 non-null   int64
 7   Obesity                          8763 non-null   int64
 8   Alcohol Consumption              8763 non-null   int64
 9   Exercise Hours Per Week          8763 non-null   float64
 10  Diet                             8763 non-null   object
 11  Medication Use                   8763 non-null   int64
 12  Stress Level                     8763 non-null   int64
 13  Sedentary Hours Per Day          8763 non-null   float64
```

```
 14  Physical Activity Days Per Week  8763 non-null   int64
 15  Sleep Hours Per Day              8763 non-null   int64
 16  Heart Attack Risk                8763 non-null   int64
dtypes: float64(2), int64(12), object(3)
memory usage: 1.1+ MB
None
```

Statistical Summary of numerical features

```
print(df.describe())
```

```
               Age  Cholesterol   Heart Rate     Diabetes      Smoking
count  8763.000000  8763.000000  8763.000000  8763.000000  8763.000000
mean     53.707977   259.877211    75.021682     0.652288     0.896839
std      21.249509    80.863276    20.550948     0.476271     0.304186
min      18.000000   120.000000    40.000000     0.000000     0.000000
25%      35.000000   192.000000    57.000000     0.000000     1.000000
50%      54.000000   259.000000    75.000000     1.000000     1.000000
75%      72.000000   330.000000    93.000000     1.000000     1.000000
max      90.000000   400.000000   110.000000     1.000000     1.000000

           Obesity  Alcohol Consumption  Exercise Hours Per Week  \
count  8763.000000          8763.000000              8763.000000
mean      0.501426             0.598083                10.014284
std       0.500026             0.490313                 5.783745
min       0.000000             0.000000                 0.002442
25%       0.000000             0.000000                 4.981579
50%       1.000000             1.000000                10.069559
75%       1.000000             1.000000                15.050018
max       1.000000             1.000000                19.998709

       Medication Use  Stress Level  Sedentary Hours Per Day  \
count     8763.000000   8763.000000              8763.000000
mean         0.498345      5.469702                 5.993690
std          0.500026      2.859622                 3.466359
min          0.000000      1.000000                 0.001263
25%          0.000000      3.000000                 2.998794
50%          0.000000      5.000000                 5.933622
75%          1.000000      8.000000                 9.019124
max          1.000000     10.000000                11.999313
```

	Physical Activity Days Per Week	Sleep Hours Per Day	Heart Atta
count	8763.000000	8763.000000	8763
mean	3.489672	7.023508	0
std	2.282687	1.988473	0
min	0.000000	4.000000	0
25%	2.000000	5.000000	0
50%	3.000000	7.000000	0
75%	5.000000	9.000000	1
max	7.000000	10.000000	1

DataFrame Information Understanding

From checking the basic dataset information, we found that as for preprocessing and cleaning, several columns may require attention. For instance, 'Age', 'Cholesterol', 'Heart Rate', 'Stress Level', 'Sedentary Hours Per Day', 'Physical Activity Days Per Week', and 'Sleep Hours Per Day' are numerical and may need to be normalized or standardized to ensure consistent scale, Additionally, the 'Diet' column, being a categorical variable with multiple categories, might require encoding (like one-hot encoding) to convert it into a numerical format suitable for analysis.

Data Cleaning and preprocessing

Data cleaning

The first step in the data cleaning process involves identifying and addressing any missing values within the dataset. Missing data can significantly impact the accuracy and reliability of the analysis. Therefore, we employed the following strategy:

1. Identification of Missing Values: Utilize pandas functions to detect any missing or null values in the dataset.
2. Decision Strategy: Based on the nature and quantity of the missing values, decide whether to impute these values or to remove the corresponding entries from the dataset.

After addressing missing values, the next step is to streamline the dataset by removing columns that are not relevant to our analysis. In this case, the 'Patient ID' column is deemed unnecessary for the following reasons:

1. Lack of Analytical Value: The ‘Patient ID’ is a unique identifier for each patient and does not contribute to the analysis of heart attack risks.
2. Data Anonymization: Removing ‘Patient ID’ ensures the privacy and anonymity of the dataset’s subjects.

Checking the missing values

```
print("Missing Values:")
print(df.isnull().sum())
```

```
Missing Values:
Patient ID                        0
Age                               0
Sex                               0
Cholesterol                       0
Heart Rate                        0
Diabetes                          0
Smoking                           0
Obesity                           0
Alcohol Consumption               0
Exercise Hours Per Week           0
Diet                              0
Medication Use                    0
Stress Level                      0
Sedentary Hours Per Day           0
Physical Activity Days Per Week   0
Sleep Hours Per Day               0
Heart Attack Risk                 0
dtype: int64
```

Droping rows with missing values if it has

```
df = df.dropna()
```

Droping the column ‘Patient ID’

```
df = df = df.drop(columns=['Patient ID'])
```

Displaying cleaned dataset

```python
print("Cleaned Dataset:\n")
print(df.head())
```

```
Cleaned Dataset:

   Age     Sex  Cholesterol  Heart Rate  Diabetes  Smoking  Obesity  \
0   67    Male          208          72         0        1        0
1   21    Male          389          98         1        1        1
2   21  Female          324          72         1        0        0
3   84    Male          383          73         1        1        0
4   66    Male          318          93         1        1        1

   Alcohol Consumption  Exercise Hours Per Week       Diet  Medication
0                    0                 4.168189    Average
1                    1                 1.813242  Unhealthy
2                    0                 2.078353    Healthy
3                    1                 9.828130    Average
4                    0                 5.804299  Unhealthy

   Stress Level  Sedentary Hours Per Day  Physical Activity Days Per We
0             9                 6.615001
1             1                 4.963459
2             9                 9.463426
3             9                 7.648981
4             6                 1.514821

   Sleep Hours Per Day  Heart Attack Risk
0                    6                  0
1                    7                  0
2                    4                  0
3                    4                  0
4                    5                  0
```

Data preprocessing

In the dataset, certain columns such as 'Sex' and 'Diet' contain categorical data. For the purposes of statistical modeling and analysis, it is essential to convert these categorical variables into a numerical format. This conversion ensures compatibility with various data analysis and machine learning algorithms which typically require numerical input.

Column 'Sex': The 'Sex' column categorizes patients into 'Male' and 'Female'. This categorical data will be converted into a binary numerical format, where one category ('Male') is represented by 1, and the other ('Female') is represented by 0.

Column 'Diet': The 'Diet' column includes categories such as 'Healthy', 'Average', and 'Unhealthy'.

Mapping 'Male' to 1 and 'Female' to 0 in the 'Sex' column

```
sex_mapping = {'Male': 1, 'Female': 0}
df['Sex'] = df['Sex'].map(sex_mapping)
```

Map 'Unhealthy' to 0, 'Average' to 1, and 'Healthy' to 2 in the 'Diet' column

```
diet_mapping = {'Unhealthy': 0, 'Average': 1, 'Healthy': 2}
df['Diet'] = df['Diet'].map(diet_mapping)
```

Exploratory Data Analysis

Exploratory Data Analysis is an integral part of any data science project, providing initial insights and guiding subsequent analysis. In this study, our EDA consists of constructing histograms for continuous data and bar charts for categorical data. This approach helps in visualizing the distribution and frequency of variables in the dataset.

Histograms for Continuous Data

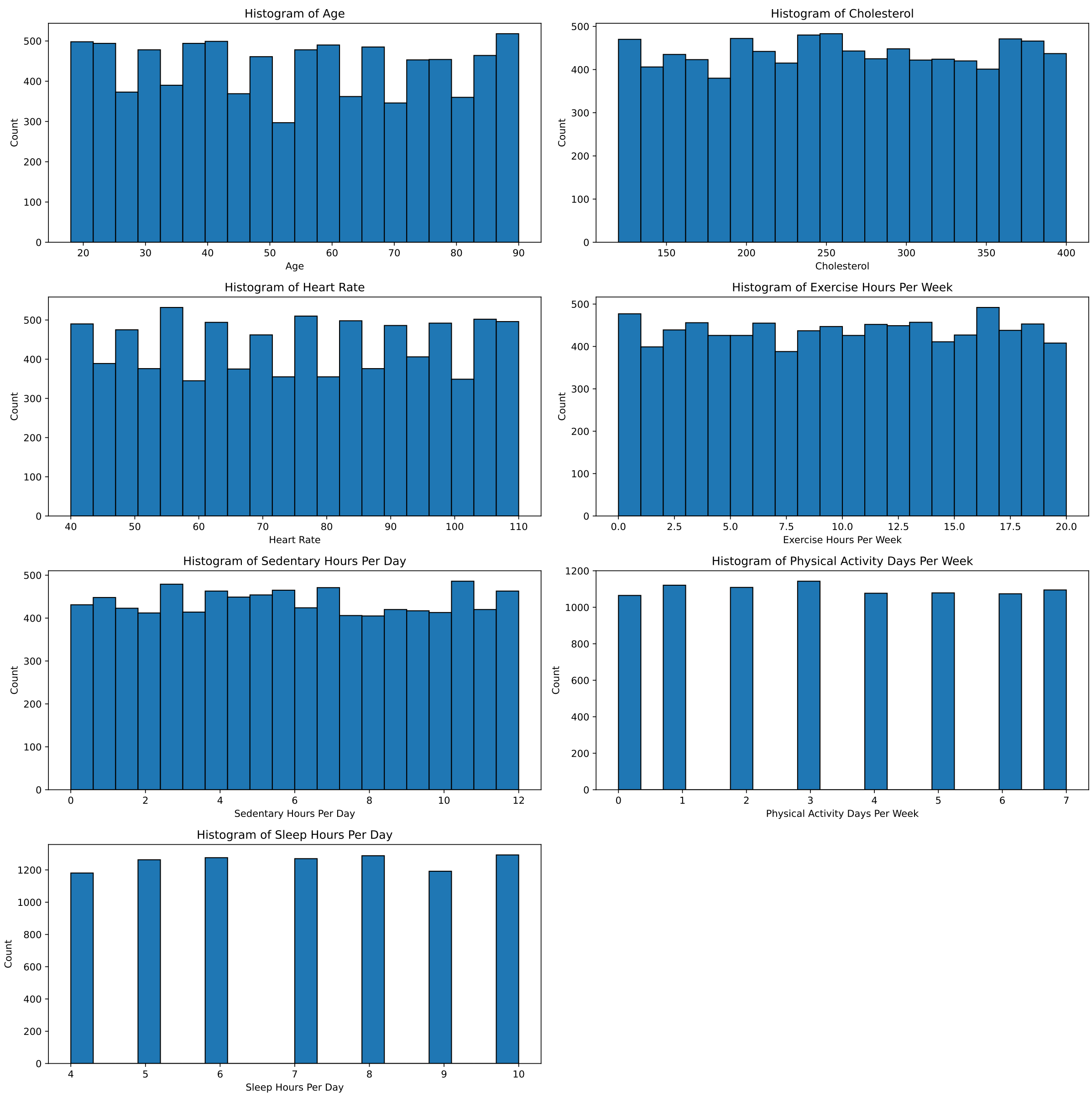
Histograms are effective in illustrating the distribution of continuous variables. They provide a visual representation of the data's spread and central tendency, and can highlight outliers or skewness in the data. For our dataset, histograms will be generated for the following continuous variables:

Age Cholesterol Heart Rate Exercise Hours Per Week Sedentary Hours Per Day Physical Activity Days Sleep Hours Per Day

Histogram Ploting

```
# List of numerical variables

numerical_vars = ['Age', 'Cholesterol', 'Heart Rate', 'Exercise Hours
                  'Sedentary Hours Per Day', 'Physical Activity Days
```

```
                  'Sleep Hours Per Day']
# Number of columns for subplots
n_cols_num = 2

# Calculate number of rows needed
n_rows_num = int(len(numerical_vars) / n_cols_num) + (len(numerical_v

# Set up the matplotlib figure for numerical variables
fig, axes = plt.subplots(n_rows_num, n_cols_num, figsize=(8 * n_cols_

# Flatten axes array for easy iteration
axes = axes.flatten()

# Loop through the numerical variables and create a histogram for eac
for i, col in enumerate(numerical_vars):
    axes[i].hist(df[col], bins=20, edgecolor='black')
    axes[i].set_title(f'Histogram of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Count')

# Remove any unused subplots
for j in range(i+1, len(axes)):
    fig.delaxes(axes[j])

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



Histograms interpretations:

1. Age: Uniformly spread, slight periodic peaks.
2. Cholesterol: Roughly normal, right-skewed indicating higher values.
3. Heart Rate: Near normal, few high values.
4. Exercise Hours: Left-skewed, most exercise little.
5. Sedentary Hours: Broad spread, regular peaks.
6. Physical Activity Days: Peaks at 0, 3, and 5-7 days suggest varied activity levels.

7. Sleep Hours: Bimodal, peaks at 6 and 8 hours, common sleep durations.

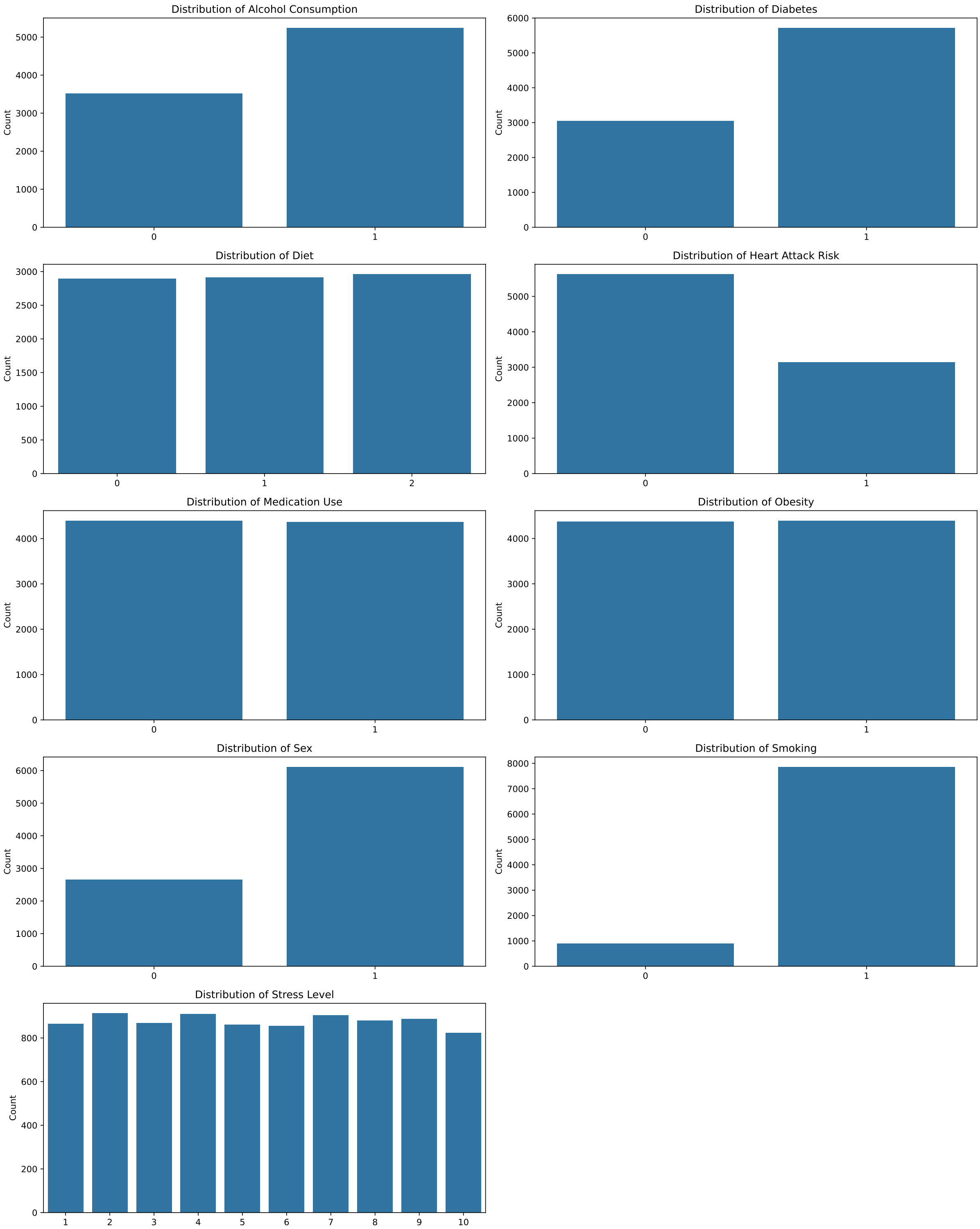
Bar Charts for Categorical Data

Bar charts are ideal for displaying the frequency distribution of categorical variables. They offer a clear view of how different categories compare in terms of frequency or count. In our dataset, bar charts will be created for the following categorical variables:

Sex Diabetes Smoking Obesity Alcohol Consumption Diet Medication Use Stress Level Physical Activity Days Per Week Heart Attack Risk

```
# Exclude the numerical variables and identifiers to get the categori
excluded_vars = numerical_vars + ['Patient ID']
categorical_vars = df.columns.difference(excluded_vars)

# Number of columns for subplots
n_cols_cat = 2

# Calculate number of rows needed
n_rows_cat = int(len(categorical_vars) / n_cols_cat) + (len(categoric

# Set up the matplotlib figure for categorical variables
fig, axes = plt.subplots(n_rows_cat, n_cols_cat, figsize=(8 * n_cols_

# Flatten axes array for easy iteration
axes = axes.flatten()

# Loop through the categorical variables and create a bar chart for e
for i, col in enumerate(categorical_vars):
    sns.countplot(x=col, data=df, ax=axes[i])
    axes[i].set_title(f'Distribution of {col}')
    axes[i].set_xlabel('')
    axes[i].set_ylabel('Count')

# Remove any unused subplots
for j in range(i+1, len(axes)):
    fig.delaxes(axes[j])

# Adjust layout to prevent overlap
plt.tight_layout()
```

```python
plt.show()
```



Bar Charts interpretation:

1. Alcohol Consumption: More individuals do consume alcohol compared to those who do not.

2. Diabetes: A larger number of individuals do have diabetes.
3. Diet: The distribution is even across diet categories, suggesting a balance between different diet types.
4. Heart Attack Risk: Fewer individuals are at risk of heart attack compared to those not at risk.
5. Medication Use: Medication Use and Obesity evenly distributed across three categories. .
6. Obesity: More individuals are obese than those who are not.
7. Sex: The distribution between genders is roughly even.
8. Smoking: Fewer individuals smoke compared to those who do not smoke.
9. Stress Level: Stress levels are evenly distributed across the scale from 1 to 10.

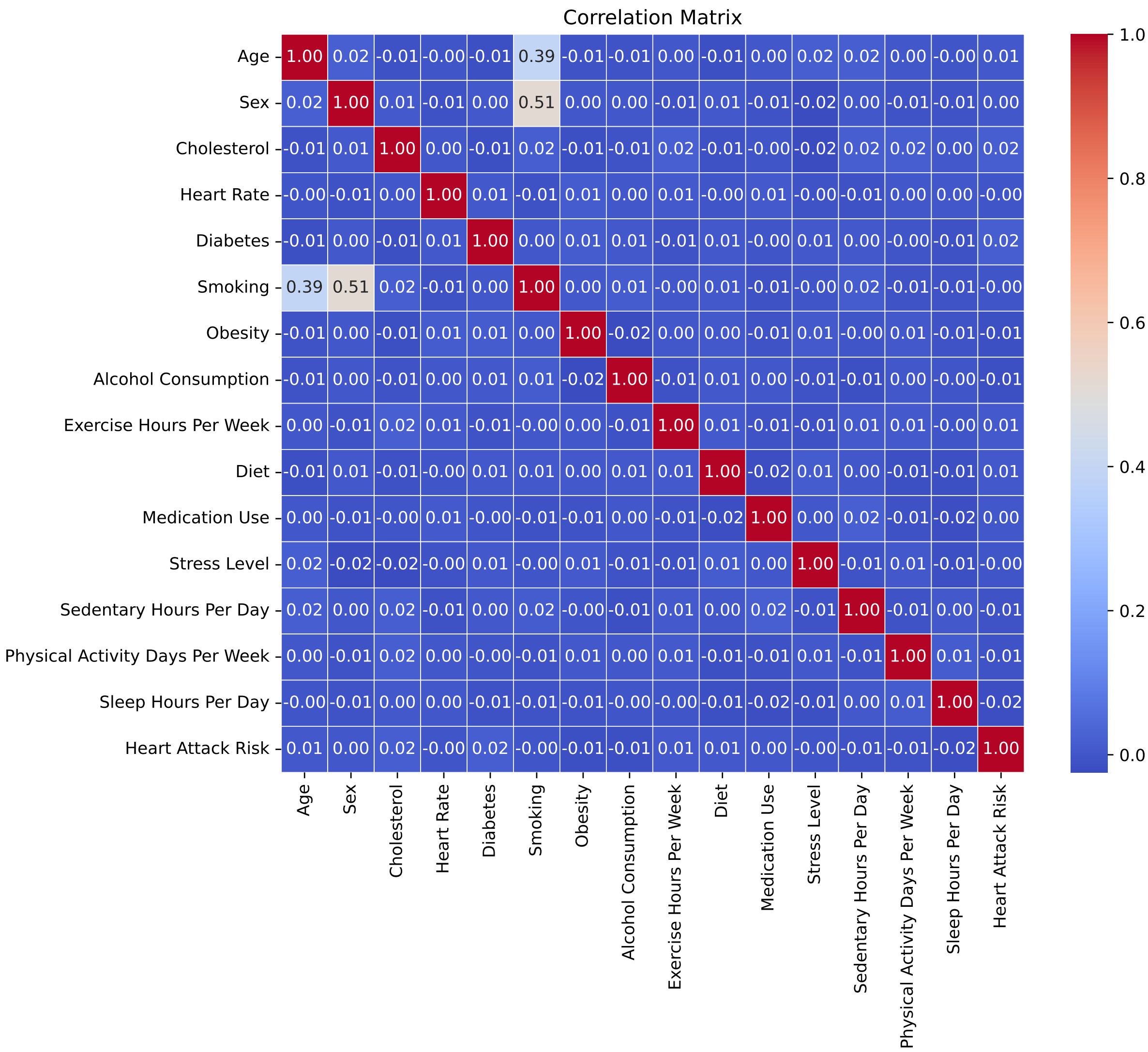
Correlation Matrix for continuous data

Correlation Matrix Graph

To gain deeper insights into the relationships between various features in the dataset, a correlation matrix graph was constructed. This graph highlights the strength and direction of the relationships between numerical variables.

Correlation Matrix

```
df_numeric = df.select_dtypes(include=['float64', 'int64'])

# Calculate the correlation matrix
matrix_correlation = df_numeric.corr()

# Use Seaborn's heatmap to plot the correlation matrix.
plt.figure(figsize=(10, 8))
sns.heatmap(matrix_correlation, annot=True, cmap='coolwarm', fmt=".2f
plt.title('Correlation Matrix')
plt.show()
```



Highest Correlation Coefficients

To determine the strongest predictors of heart attack risk, the correlation coefficients of each feature with the 'Heart Attack Risk' variable were analyzed.

```
df_numeric = df.select_dtypes(include=['float64', 'int64'])

# Calculate the correlation matrix
matrix_correlation = df_numeric.corr()

# Determine the relationship between each characteristic and the targ
target_correlation = matrix_correlation['Heart Attack Risk'].abs()

# Sort the characteristics according to their relationship to the tar
```

```python
sorted_correlation = target_correlation.sort_values(ascending=False)

# Print the features with the highest correlation coefficients
print("Features with the highest correlation in descending order:")
print(sorted_correlation)

# Choose the top five correlated features.
top5_features = sorted_correlation.index[1:6]

# Display the top 5 correlated features
print("Top 5 Correlated Features with Heart Attack Risk:\n")
print(top5_features)
```

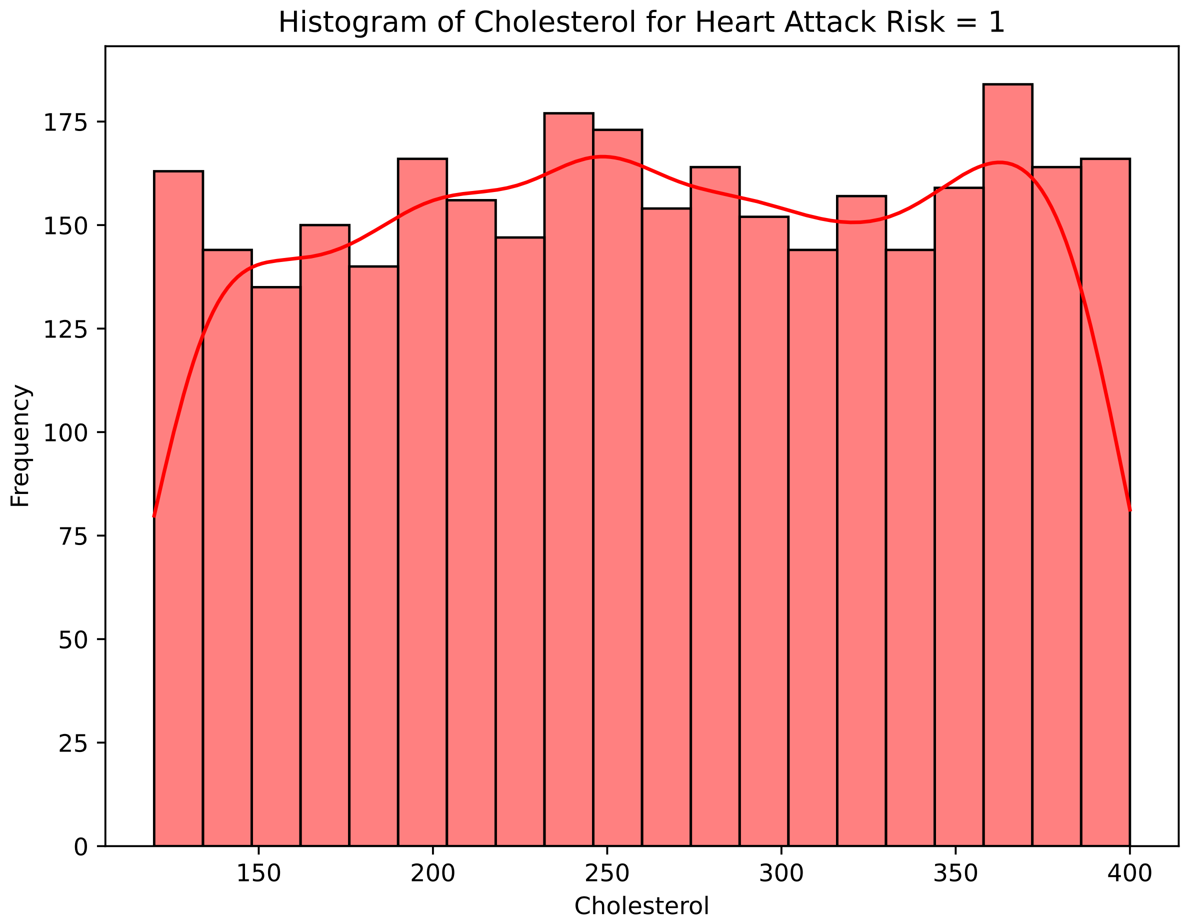
```
Features with the highest correlation in descending order:
Heart Attack Risk                  1.000000
Cholesterol                        0.019340
Sleep Hours Per Day                0.018528
Diabetes                           0.017225
Alcohol Consumption                0.013778
Obesity                            0.013318
Exercise Hours Per Week            0.011133
Age                                0.006403
Diet                               0.005908
Sedentary Hours Per Day            0.005613
Physical Activity Days Per Week    0.005014
Heart Rate                         0.004251
Stress Level                       0.004111
Smoking                            0.004051
Sex                                0.003095
Medication Use                     0.002234
Name: Heart Attack Risk, dtype: float64
Top 5 Correlated Features with Heart Attack Risk:

Index(['Cholesterol', 'Sleep Hours Per Day', 'Diabetes', 'Alcohol Consu
       'Obesity'],
      dtype='object')
```

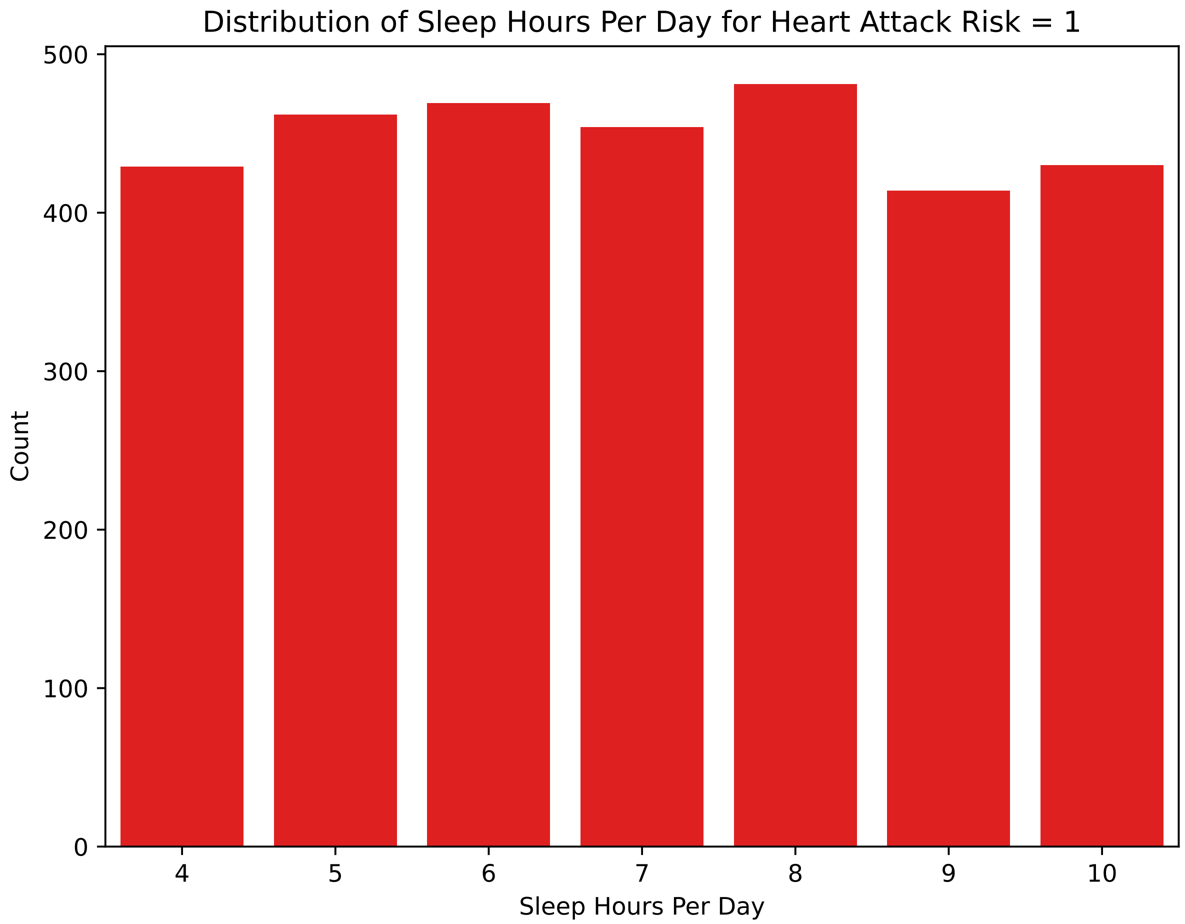
Analysis of Top 5 Correlated Features

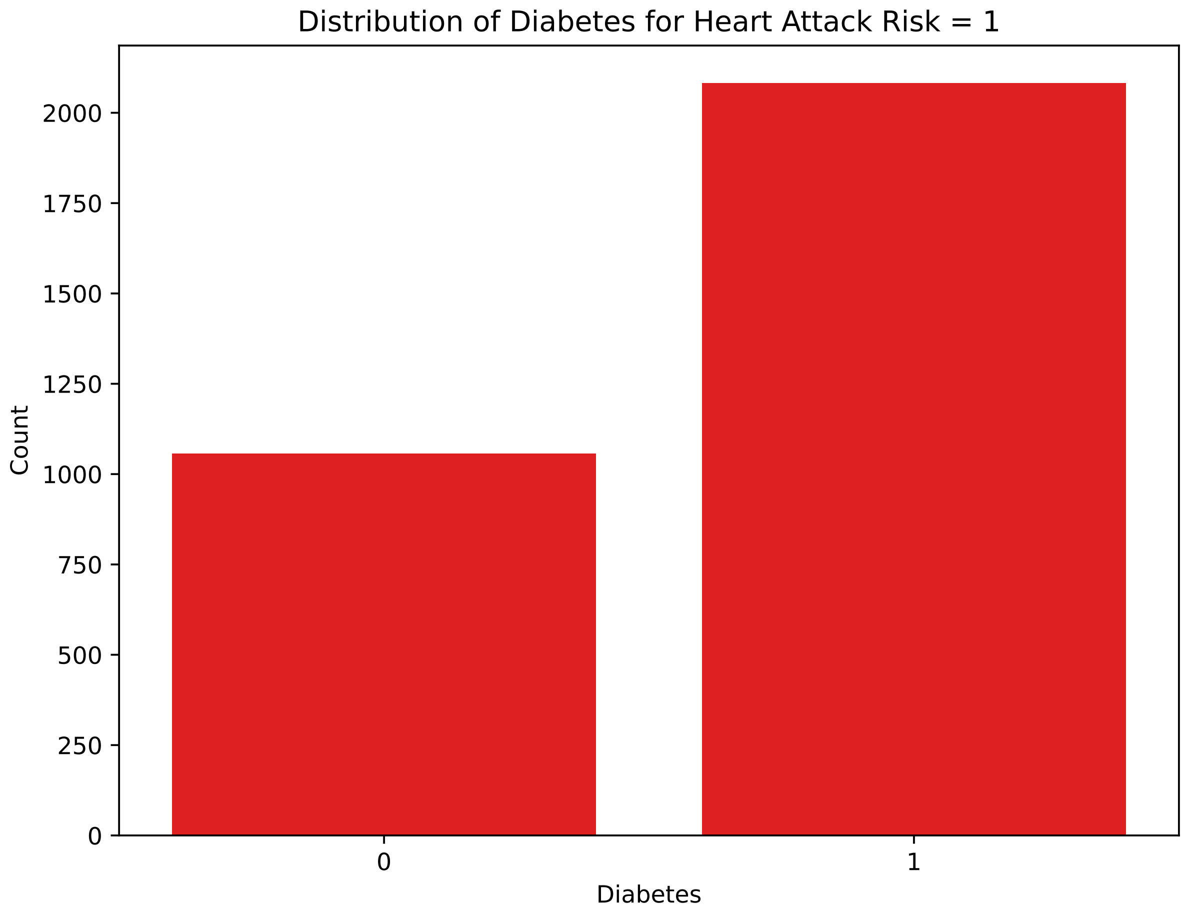
A focused analysis was conducted on the top five features most strongly correlated with heart attack risk. Histograms or bar plots were created for these features, specifically for patients with a heart attack risk.Top 5 features that causes heart attack and are highly correlated Cholesterol Sleep Hours Diabetes Alcohol Consumption Obesity

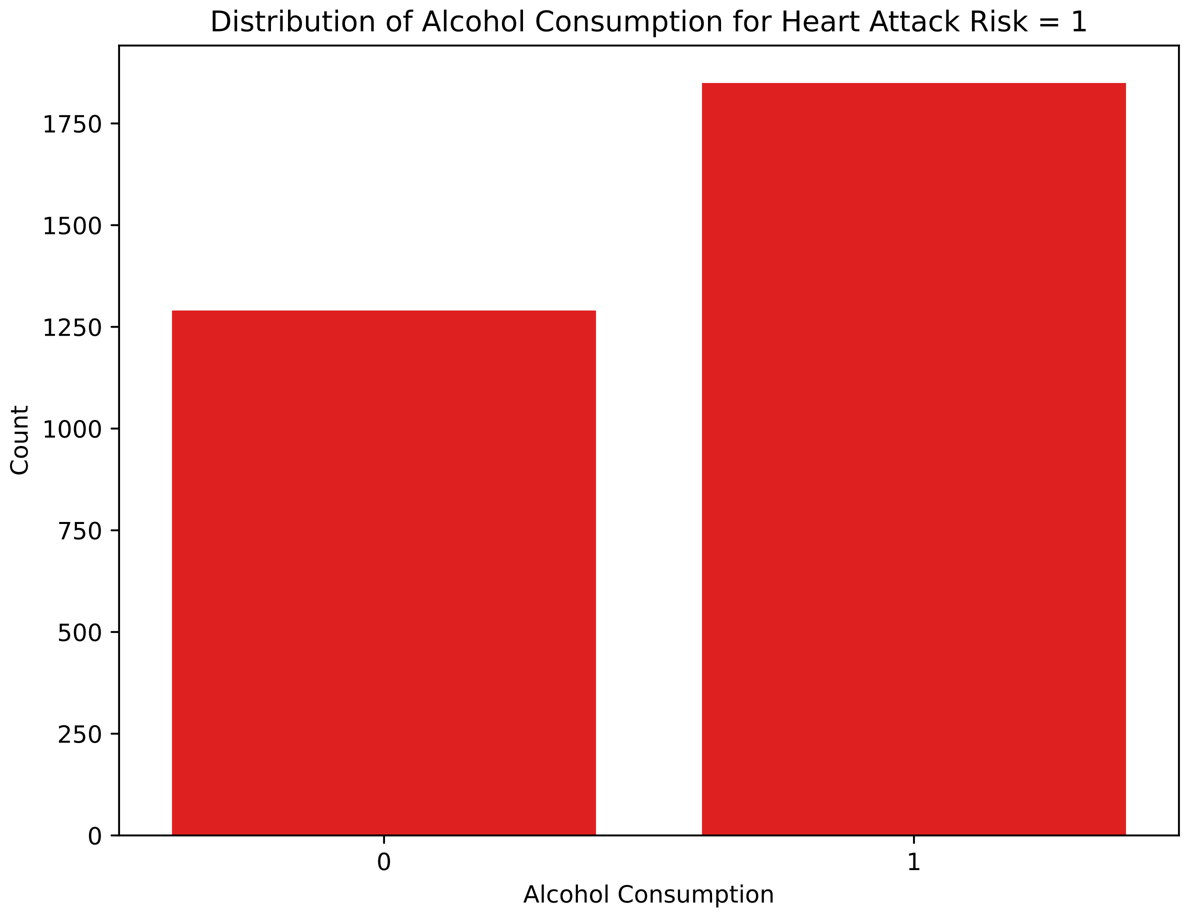
```
# Choose the top five correlated features.
top5_features = sorted_correlation.index[1:6]

# Display the top 5 correlated features
print("Top 5 Correlated Features with Heart Attack Risk:\n")
print(top5_features)


# Filter rows where "Heart Attack Risk" is 1
heart_attack_df = df[df['Heart Attack Risk'] == 1]

top5_features = sorted_correlation.index[1:6]  # Exclude 'Heart Attac

# Plot histograms or bar plots for the top 5 correlated features
for feature in top5_features:
    if df[feature].dtype == 'object' or df[feature].nunique() <= 10:
        plt.figure(figsize=(8, 6))
        sns.countplot(x=feature, data=heart_attack_df, color='red')
        plt.title(f'Distribution of {feature} for Heart Attack Risk =
        plt.xlabel(feature)
        plt.ylabel('Count')
    else:  # Continuous data
        plt.figure(figsize=(8, 6))
        sns.histplot(heart_attack_df[feature], bins=20, kde=True, col
        plt.title(f'Histogram of {feature} for Heart Attack Risk = 1'
        plt.xlabel(feature)
        plt.ylabel('Frequency')
    plt.show()
```

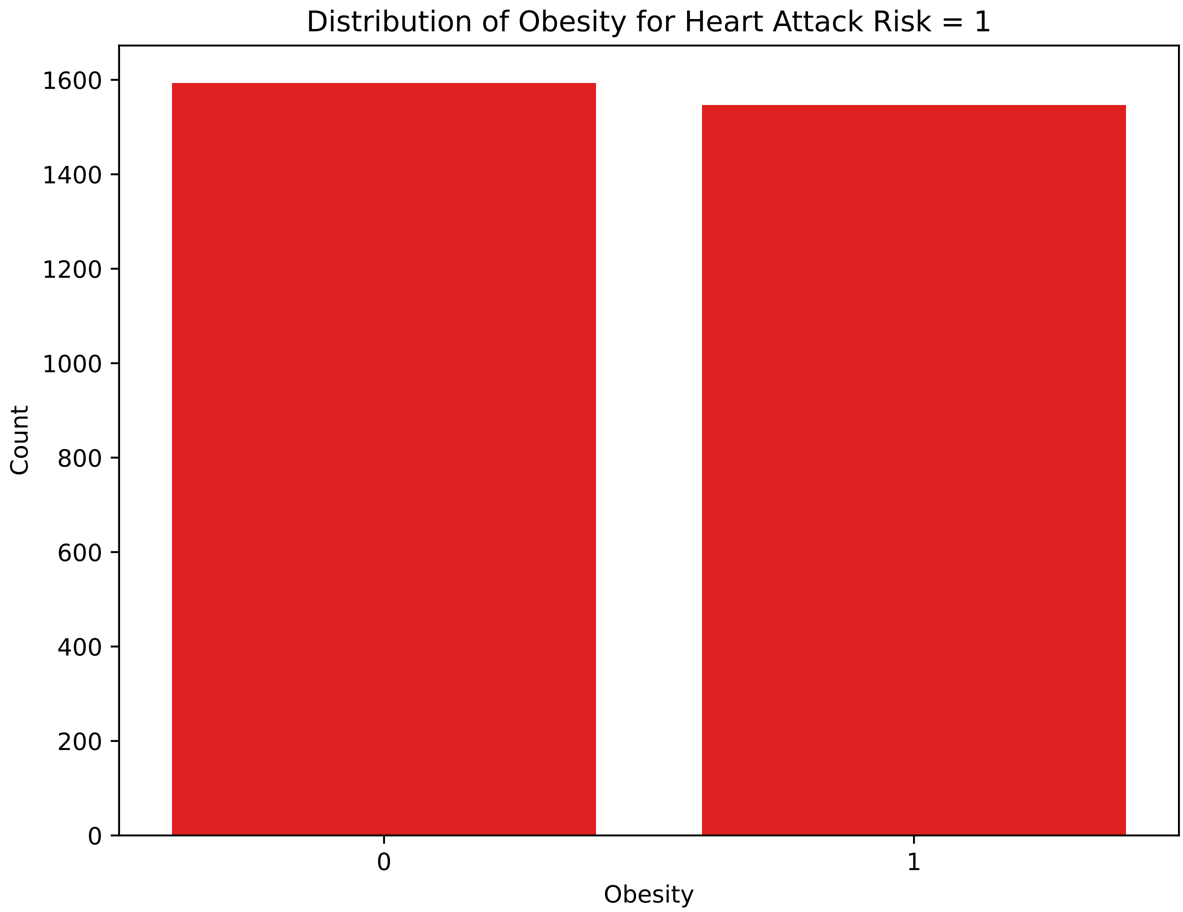
```
Top 5 Correlated Features with Heart Attack Risk:

Index(['Cholesterol', 'Sleep Hours Per Day', 'Diabetes', 'Alcohol Consu
       'Obesity'],
      dtype='object')
```


Histogram of Cholesterol for Heart Attack Risk = 1
Frequency
Cholesterol
0
25
50
75
100
125
150
175
150
200
250
300
350
400


Distribution of Sleep Hours Per Day for Heart Attack Risk = 1
Count
0
100
200
300
400
500
4
5
6
7
8
9
10
Sleep Hours Per Day


Distribution of Diabetes for Heart Attack Risk = 1
Count
2000
1750
1500
1250
1000
750
500
250
0
0
1
Diabetes


Distribution of Alcohol Consumption for Heart Attack Risk = 1
Count
0
250
500
750
1000
1250
1500
1750
0
1
Alcohol Consumption



Correlation Matrix for continuous data interpretations:

1. Cholesterol Histogram: The distribution of cholesterol among high-risk individuals shows variability with a range of peaks, indicating no single dominant cholesterol level associated with increased heart attack risk.

2. Sleep Hours Histogram: Sleep duration for high-risk individuals is fairly evenly distributed from 4 to 9 hours, with no specific sleep duration appearing to be significantly more common in this group.

3. Diabetes Histogram: A greater number of individuals at high risk for heart attacks are diabetic, with diabetic individuals outnumbering non-diabetics almost 2 to 1.

4. Alcohol Consumption Histogram: More individuals at high risk for heart attacks consume alcohol than do not, suggesting a potential link between alcohol consumption and increased heart attack risk.

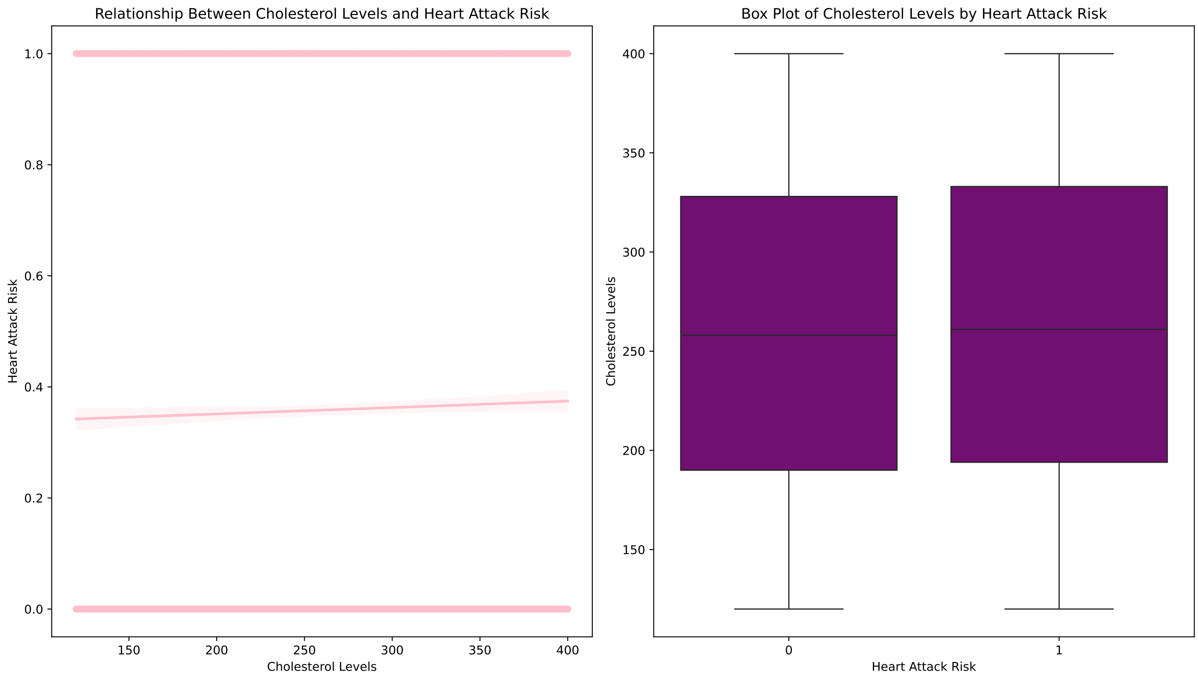
5. Obesity Histogram: The distribution between obese and non-obese individuals in the high-risk category is nearly even, suggesting obesity is a common trait among those at high risk for heart attacks.

In-depth Feature Analysis with Heart Attack Risk

The Relationship Between Cholesterol and Heart Attack Risk

Following the exploratory data analysis, we focused on examining the relationship between cholesterol levels and heart attack risk.

```
plt.figure(figsize=(14, 8))
# Creating a subplot for the regplot
plt.subplot(1, 2, 1)
sns.regplot(x='Cholesterol', y='Heart Attack Risk', data=df, logistic
plt.title('Relationship Between Cholesterol Levels and Heart Attack R
plt.xlabel('Cholesterol Levels')
plt.ylabel('Heart Attack Risk')

# Creating a subplot for the boxplot
plt.subplot(1, 2, 2)
sns.boxplot(x='Heart Attack Risk', y='Cholesterol', data=df, color='p
plt.title('Box Plot of Cholesterol Levels by Heart Attack Risk')
plt.xlabel('Heart Attack Risk')
plt.ylabel('Cholesterol Levels')

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



Interpretations:

1. Regression Plot (Regplot): A logistic regression plot was generated to visualize the relationship between cholesterol levels and heart attack risk. This plot indicates a positive correlation, suggesting that higher cholesterol levels may be associated with an increased risk of heart attacks.

2. Box Plot: Additionally, a box plot was created to compare cholesterol levels across different heart attack risk categories. The plot shows that patients with a heart attack risk of 1 tend to have higher cholesterol levels.

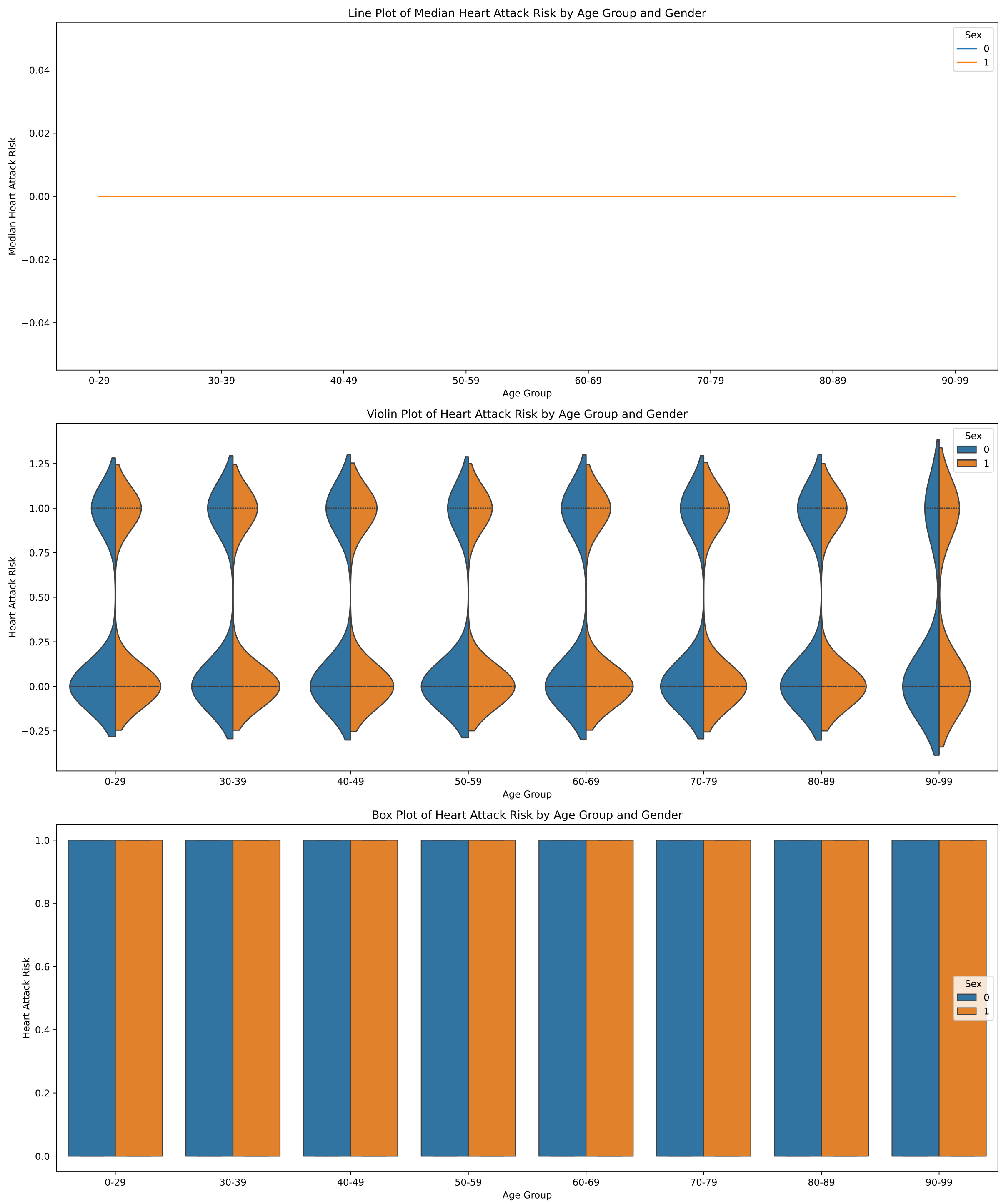
In the 1st regplot, it shows Cholesterol has a positive affect on Heart Attack Risk. In the 2nd boxplot, the Heart Attack Risk =1 looks like to have a higher Cholesterol looks.

The Relationship Between Age or Sex and Heart Attack Risk

To further investigate the potential influence of age and sex on heart attack risk, various plots were created after defining age groups.

```
# Define age groups
age_bins = [0, 29, 39, 49, 59, 69, 79, 89, 99]
age_labels = ['0-29', '30-39', '40-49', '50-59', '60-69', '70-79', '8
df['Age Group'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels,
```

```
# Create a subplot layout
fig, axes = plt.subplots(3, 1, figsize=(15, 18))

# Create a line plot for the median heart attack risk by age group an
sns.lineplot(data=df, x='Age Group', y='Heart Attack Risk', hue='Sex'
axes[0].set_title('Line Plot of Median Heart Attack Risk by Age Group
axes[0].set_xlabel('Age Group')
axes[0].set_ylabel('Median Heart Attack Risk')

# Create a violin plot for heart attack risk by age group and gender
sns.violinplot(data=df, x='Age Group', y='Heart Attack Risk', hue='Se
axes[1].set_title('Violin Plot of Heart Attack Risk by Age Group and
axes[1].set_xlabel('Age Group')
axes[1].set_ylabel('Heart Attack Risk')

# Create a box plot for heart attack risk by age group and gender
sns.boxplot(data=df, x='Age Group', y='Heart Attack Risk', hue='Sex',
axes[2].set_title('Box Plot of Heart Attack Risk by Age Group and Gen
axes[2].set_xlabel('Age Group')
axes[2].set_ylabel('Heart Attack Risk')

# Adjust the layout
plt.tight_layout()
plt.show()
```



Interpretations:

1. Line Plot: This plot showed the median heart attack risk by age group and gender, indicating that age and sex alone may not be strong individual predictors of heart attack risk.
2. Violin and Box Plots: These plots provided a deeper view of heart

attack risk distribution across different age groups and between genders, reinforcing the findings of the line plot.

From the above plots, we can see that sex and age cannot be the strong individual predictors of heart attack risk.

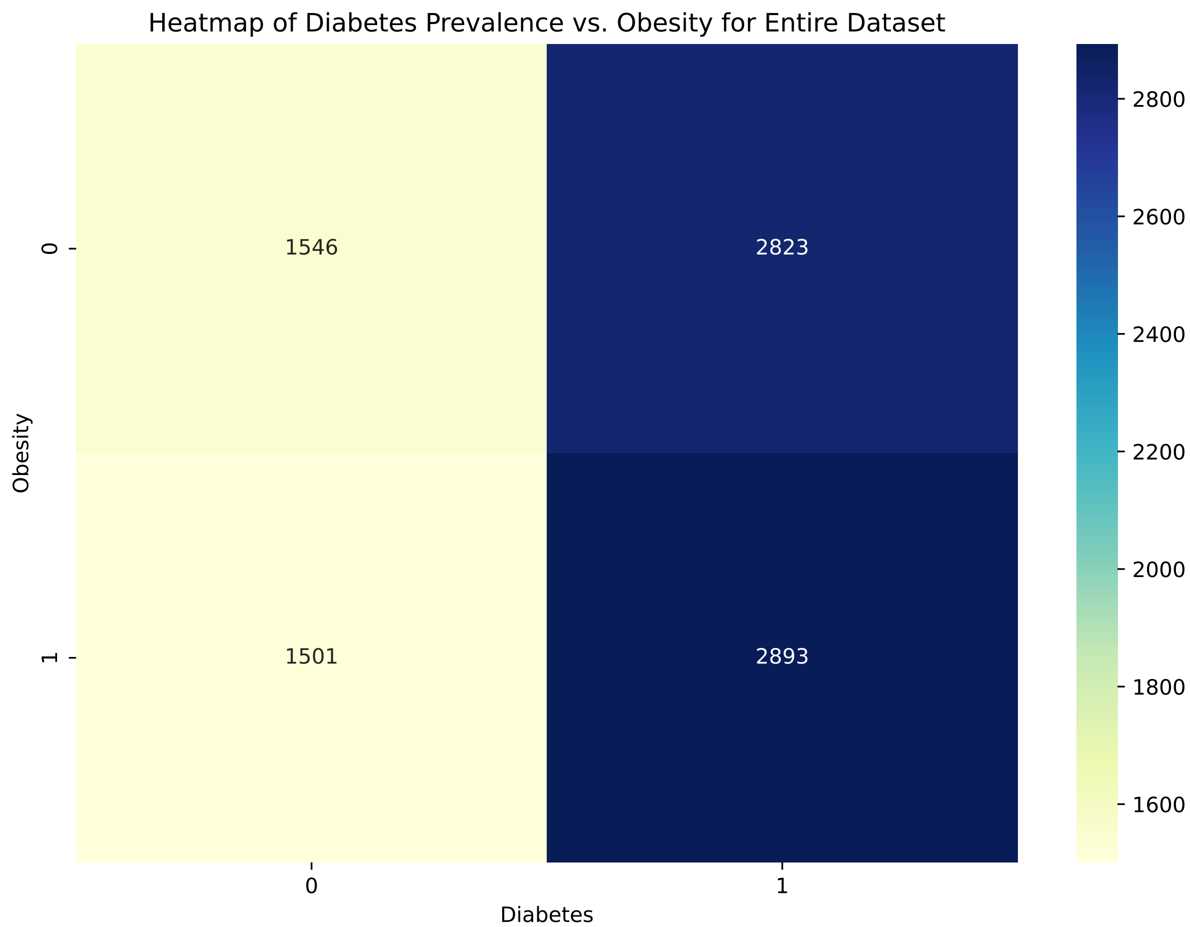
Analyzing Diabetes Prevalence in Obese vs. Non-Obese by Age

An in-depth analysis was conducted to explore the relationship between obesity, diabetes, and age.

Create a contingency table for obesity and diabetes for the entire dataset

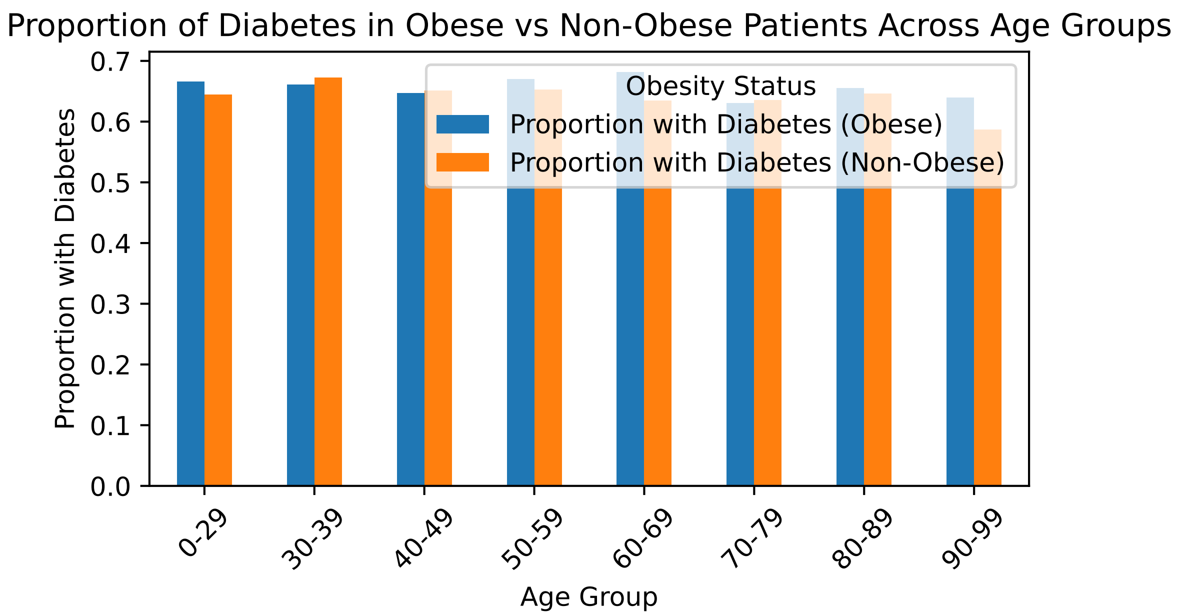
```
contingency_table_all = pd.crosstab(df['Obesity'], df['Diabetes'])

# Plotting the contingency table as a heatmap
plt.figure(figsize=(10, 7))
sns.heatmap(contingency_table_all, annot=True, fmt='d', cmap="YlGnBu"
plt.title('Heatmap of Diabetes Prevalence vs. Obesity for Entire Data
plt.xlabel('Diabetes')
plt.ylabel('Obesity')
plt.show()

# Calculate proportions of diabetes for obese and non-obese in each a
age_group_proportions = df.groupby('Age Group').apply(
    lambda x: pd.Series({
        'Proportion with Diabetes (Obese)': x[x['Obesity'] == 1]['Dia
        'Proportion with Diabetes (Non-Obese)': x[x['Obesity'] == 0][
    })
).reset_index()

# Plotting the proportions as a bar chart
plt.figure(figsize=(14, 7))
age_group_proportions.set_index('Age Group').plot(kind='bar', stacked
plt.title('Proportion of Diabetes in Obese vs Non-Obese Patients Acro
plt.xlabel('Age Group')
plt.ylabel('Proportion with Diabetes')
plt.xticks(rotation=45)
plt.legend(title='Obesity Status')
plt.tight_layout()
plt.show()
```

```python
from scipy.stats import chi2_contingency

# Create a contingency table for obesity and diabetes for the entire
contingency_table_all = pd.crosstab(df['Obesity'], df['Diabetes'])

# Perform the chi-squared test for the entire dataset
chi2, p_value_all, dof, expected = chi2_contingency(contingency_table

# Perform the chi-squared test for each age group and store results
age_group_results = []
for group in age_labels:
    age_group_data = df[df['Age Group'] == group]
    contingency_table_age_group = pd.crosstab(age_group_data['Obesity
    chi2_age, p_value_age, dof_age, expected_age = chi2_contingency(c
    age_group_results.append((group, chi2_age, p_value_age))

# Display results
contingency_table_all, chi2, p_value_all, age_group_results
```




```
C:\Users\HP\AppData\Local\Temp\ipykernel_13304\4067109074.py:12: Future
  age_group_proportions = df.groupby('Age Group').apply(
```

```
<Figure size 4200x2100 with 0 Axes>
```



```
(Diabetes     0     1
 Obesity
 0          1546  2823
 1          1501  2893,
 1.396992947634959,
 0.23722769870856877,
 [('0-29', 0.6044036236112247, 0.43690275970688497),
  ('30-39', 0.12796027930816933, 0.720556336564061 3),
  ('40-49', 0.0073672768348610 7, 0.9315992983476237),
  ('50-59', 0.3017332471409519, 0.5827978653496915),
  ('60-69', 2.793185700920817, 0.0946658614866286),
  ('70-79', 0.012242787575734 83, 0.9118961718666397),
  ('80-89', 0.06294613170305828, 0.8018986327673391),
  ('90-99', 0.5948801432378257, 0.44053817808833196)])
```

Interpretations:

1. Heatmap of Diabetes Prevalence vs. Obesity: A heatmap was created to visualize the relationship between obesity and diabetes prevalence.
2. Bar Chart of Diabetes Proportions by Age Group: This chart showed the proportion of diabetes in obese versus non-obese patients across different age groups, highlighting the varying prevalence of diabetes in relation to obesity status and age.
3. Statistical Analysis: The Chi-squared test was performed to evaluate the statistical significance of the association between obesity and diabetes across all age groups.

Data Modeling

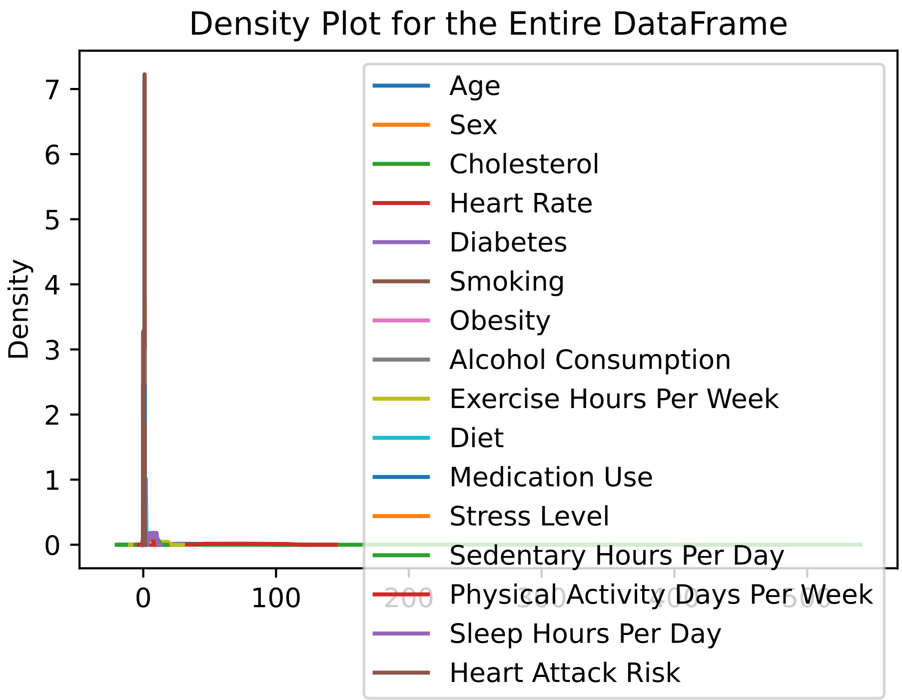
In this chapter, we delve into various modeling techniques and examine the goodness of fit model applied to the dataset, specifically focusing on predicting the risk of heart attacks.

Normalization and Standarzation of the dataset

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

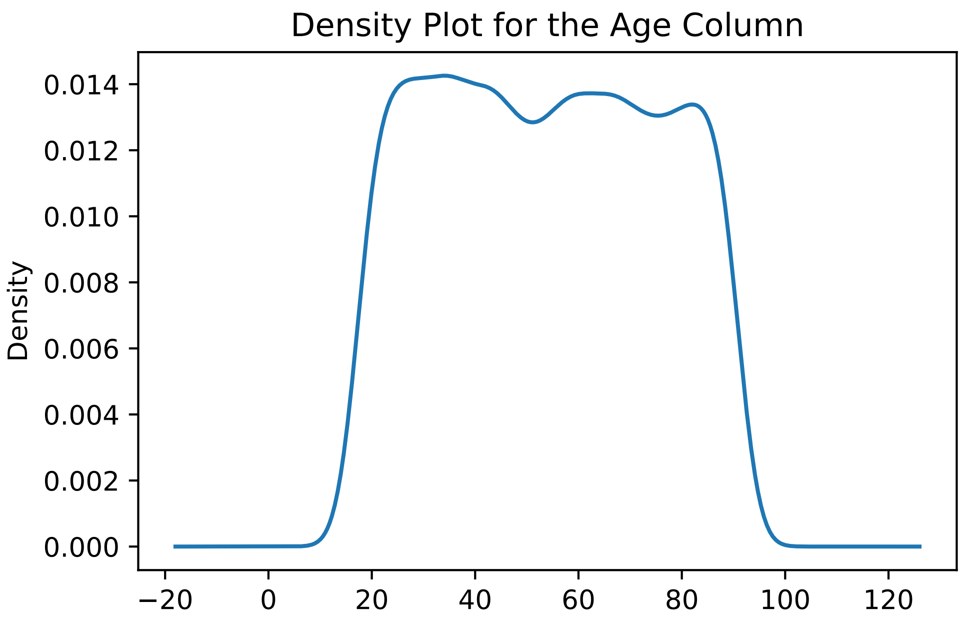
Density Plot for the entire DataFrame

```
df.plot(kind='density')
plt.title('Density Plot for the Entire DataFrame')
plt.show()
```



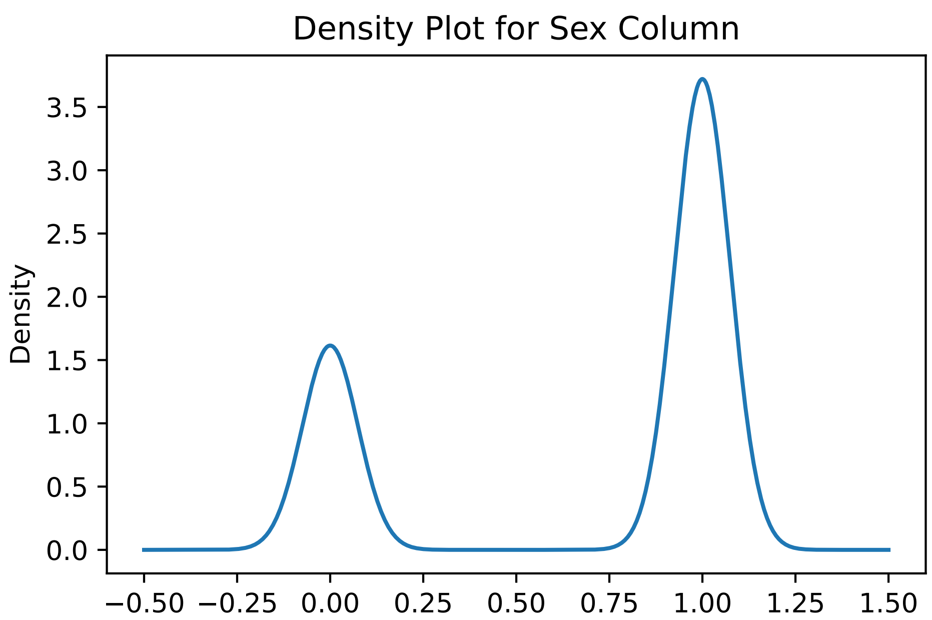
Ploting the density of the 'Age' column

```
df['Age'].plot(kind='density')
plt.title('Density Plot for the Age Column')
plt.show()
```



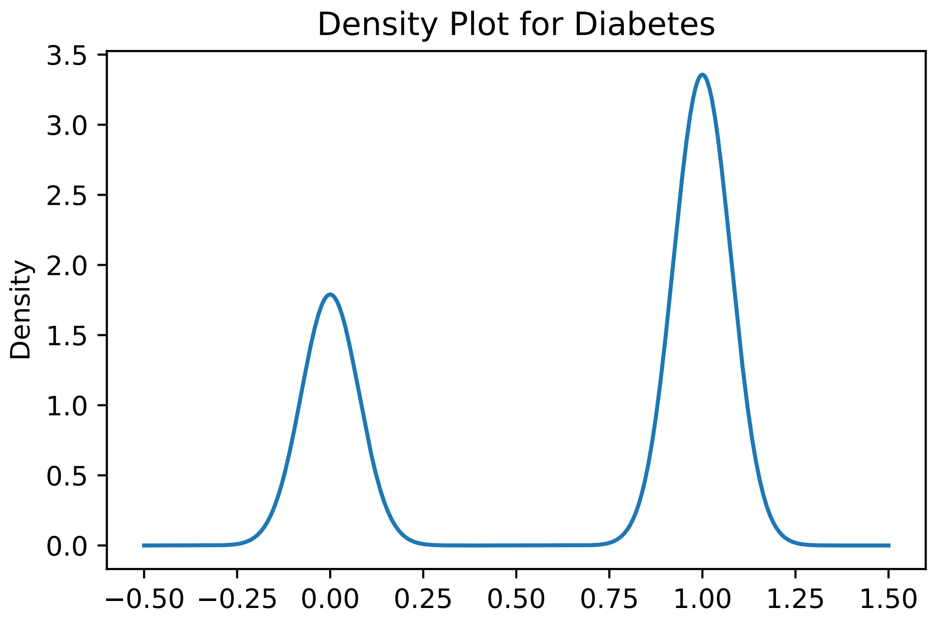
Ploting the density of the 'Sex' Column

```
df['Sex'].plot(kind='density')
plt.title("Density Plot for Sex Column")
plt.show()
```



Ploting the density of "Cholesterol" column

```
df['Diabetes'].plot(kind='density')
plt.title("Density Plot for Diabetes")
plt.show()
```



Standardization and Normalization of the dataset

```
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
from numpy import set_printoptions


X = df[['Age', 'Sex', 'Cholesterol', 'Heart Rate', 'Diabetes', 'Smoki
        'Alcohol Consumption', 'Exercise Hours Per Week', 'Diet', 'Me
        'Stress Level', 'Sedentary Hours Per Day', 'Physical Activity
        'Sleep Hours Per Day']]
y = df['Heart Attack Risk']

scaler = MinMaxScaler(feature_range=(0, 1))
rescale = scaler.fit_transform(X)  # Apply fit_transform to the featu

set_printoptions(precision=3)


# Converting it back to DataFrame
```

```
rescaleDf = pd.DataFrame(rescale, columns=['Age', 'Sex', 'Cholesterol
                                           'Diabetes', 'Smoking', 'Ob
                                           'Alcohol Consumption', 'Ex
                                           'Medication Use', 'Stress
                                           'Physical Activity Days Pe

# Adding back the y variable
rescaleDf['Heart Attack Risk'] = y

print(rescaleDf)
```

```
           Age  Sex  Cholesterol  Heart Rate  Diabetes  Smoking  Obesit
0     0.680556  1.0     0.314286    0.457143       0.0      1.0      0.
1     0.041667  1.0     0.960714    0.828571       1.0      1.0      1.
2     0.041667  0.0     0.728571    0.457143       1.0      0.0      0.
3     0.916667  1.0     0.939286    0.471429       1.0      1.0      0.
4     0.666667  1.0     0.707143    0.757143       1.0      1.0      1.
...        ...  ...          ...         ...       ...      ...      ..
8758  0.583333  1.0     0.003571    0.300000       1.0      1.0      0.
8759  0.138889  0.0     0.000000    0.471429       1.0      0.0      1.
8760  0.402778  1.0     0.464286    0.928571       0.0      1.0      1.
8761  0.250000  1.0     0.207143    0.285714       1.0      1.0      0.
8762  0.097222  0.0     0.842857    0.500000       1.0      0.0      0.

      Alcohol Consumption  Exercise Hours Per Week  Diet  Medication Us
0                     0.0                 0.208326   0.5              0.
1                     1.0                 0.090557   0.0              0.
2                     0.0                 0.103815   1.0              1.
3                     1.0                 0.491376   0.5              0.
4                     0.0                 0.290147   0.0              0.
...                   ...                      ...   ...              ..
8758                  1.0                 0.395819   1.0              1.
8759                  0.0                 0.827954   1.0              0.
8760                  1.0                 0.157329   0.5              0.
8761                  0.0                 0.189411   0.0              1.
8762                  1.0                 0.904134   1.0              0.

      Stress Level  Sedentary Hours Per Day  Physical Activity Days Per
0         0.888889                 0.551234                         0.0
1         0.000000                 0.413584                         0.1
```

```
2         0.888889                  0.788642                        0.5
3         0.888889                  0.637413                        0.4
4         0.555556                  0.126150                        0.1
...            ...                       ...
8758      0.777778                  0.900572                        1.0
8759      0.777778                  0.319366                        0.5
8760      0.444444                  0.197861                        0.5
8761      0.444444                  0.002320                        0.2
8762      0.777778                  0.750453                        1.0

      Sleep Hours Per Day  Heart Attack Risk
0                0.333333                  0
1                0.500000                  0
2                0.000000                  0
3                0.000000                  0
4                0.166667                  0
...                   ...                ...
8758             0.500000                  0
8759             0.833333                  0
8760             0.000000                  1
8761             0.666667                  0
8762             0.000000                  1

[8763 rows x 16 columns]
```

PerForming Normalization On the DataFrame

```
from sklearn.preprocessing import Normalizer
from numpy import set_printoptions
import pandas as pd

X = rescaleDf[['Age', 'Sex', 'Cholesterol', 'Heart Rate', 'Diabetes',
               'Obesity', 'Alcohol Consumption', 'Exercise Hours Per
               'Medication Use', 'Stress Level', 'Sedentary Hours Per
               'Physical Activity Days Per Week', 'Sleep Hours Per Da
y = rescaleDf['Heart Attack Risk']

scalerN = Normalizer().fit(X)
reNormalizeX = scalerN.transform(X)
```

```
set_printoptions(precision=3)
print(reNormalizeX)

# Converting it back to a DataFrame
reNormalizeDf = pd.DataFrame(reNormalizeX, columns=['Age', 'Sex', 'Ch
                                                    'Obesity', 'Alcoho
                                                    'Medication Use',
                                                    'Physical Activity


# Adding back y
reNormalizeDf['Heart Attack Risk'] = y

print(reNormalizeDf)


print(reNormalizeDf.columns)
print(reNormalizeDf.info())
```

```
[[0.329 0.484 0.152 ... 0.267 0.    0.161]
 [0.016 0.376 0.362 ... 0.156 0.054 0.188]
 [0.018 0.    0.311 ... 0.337 0.244 0.   ]
 ...
 [0.163 0.406 0.188 ... 0.08  0.232 0.   ]
 [0.112 0.45  0.093 ... 0.001 0.128 0.3  ]
 [0.037 0.    0.32  ... 0.285 0.379 0.   ]]
           Age       Sex  Cholesterol  Heart Rate  Diabetes   Smoking
0     0.329367  0.483968     0.152104    0.221242  0.000000  0.483968
1     0.015680  0.376331     0.361547    0.311817  0.376331  0.376331
2     0.017781  0.000000     0.310921    0.195087  0.426754  0.000000
3     0.327876  0.357683     0.335967    0.168622  0.357683  0.357683
4     0.272741  0.409112     0.289301    0.309756  0.409112  0.409112
...        ...       ...          ...         ...       ...       ...
8758  0.191768  0.328746     0.001174    0.098624  0.328746  0.328746
8759  0.058405  0.000000     0.000000    0.198245  0.420519  0.000000
8760  0.163375  0.405620     0.188324    0.376647  0.000000  0.405620
8761  0.112406  0.449624     0.093136    0.128464  0.449624  0.449624
8762  0.036864  0.000000     0.319589    0.189587  0.379173  0.000000

       Obesity  Alcohol Consumption  Exercise Hours Per Week      Diet
0     0.000000             0.000000                 0.100823  0.241984
```

```
1     0.376331             0.376331                  0.034079  0.000000
2     0.000000             0.000000                  0.044303  0.426754
3     0.000000             0.357683                  0.175757  0.178842
4     0.409112             0.000000                  0.118703  0.000000
...        ...                  ...                       ...       ...
8758  0.000000             0.328746                  0.130124  0.328746
8759  0.420519             0.000000                  0.348170  0.420519
8760  0.405620             0.405620                  0.063816  0.202810
8761  0.000000             0.000000                  0.085164  0.000000
8762  0.000000             0.379173                  0.342824  0.379173

      Medication Use  Stress Level  Sedentary Hours Per Day  \
0           0.000000      0.430193                 0.266780
1           0.000000      0.000000                 0.155644
2           0.426754      0.379337                 0.336556
3           0.000000      0.317941                 0.227992
4           0.000000      0.227285                 0.051610
...              ...           ...                      ...
8758        0.328746      0.255691                 0.296059
8759        0.000000      0.327070                 0.134300
8760        0.000000      0.180276                 0.080257
8761        0.449624      0.199833                 0.001043
8762        0.000000      0.294913                 0.284552

      Physical Activity Days Per Week  Sleep Hours Per Day  Heart Attac
0                            0.000000             0.161323
1                            0.053762             0.188166
2                            0.243859             0.000000
3                            0.153293             0.000000
4                            0.058445             0.068185
...                               ...                  ...
8758                         0.328746             0.164373
8759                         0.240297             0.350432
8760                         0.231783             0.000000
8761                         0.128464             0.299749
8762                         0.379173             0.000000

[8763 rows x 16 columns]
Index(['Age', 'Sex', 'Cholesterol', 'Heart Rate', 'Diabetes', 'Smoking'
       'Obesity', 'Alcohol Consumption', 'Exercise Hours Per Week', 'Di
       'Medication Use', 'Stress Level', 'Sedentary Hours Per Day',
```

```
       'Physical Activity Days Per Week', 'Sleep Hours Per Day',
       'Heart Attack Risk'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8763 entries, 0 to 8762
Data columns (total 16 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Age                              8763 non-null   float64
 1   Sex                              8763 non-null   float64
 2   Cholesterol                      8763 non-null   float64
 3   Heart Rate                       8763 non-null   float64
 4   Diabetes                         8763 non-null   float64
 5   Smoking                          8763 non-null   float64
 6   Obesity                          8763 non-null   float64
 7   Alcohol Consumption              8763 non-null   float64
 8   Exercise Hours Per Week          8763 non-null   float64
 9   Diet                             8763 non-null   float64
 10  Medication Use                   8763 non-null   float64
 11  Stress Level                     8763 non-null   float64
 12  Sedentary Hours Per Day          8763 non-null   float64
 13  Physical Activity Days Per Week  8763 non-null   float64
 14  Sleep Hours Per Day              8763 non-null   float64
 15  Heart Attack Risk                8763 non-null   int64
dtypes: float64(15), int64(1)
memory usage: 1.1 MB
None
```

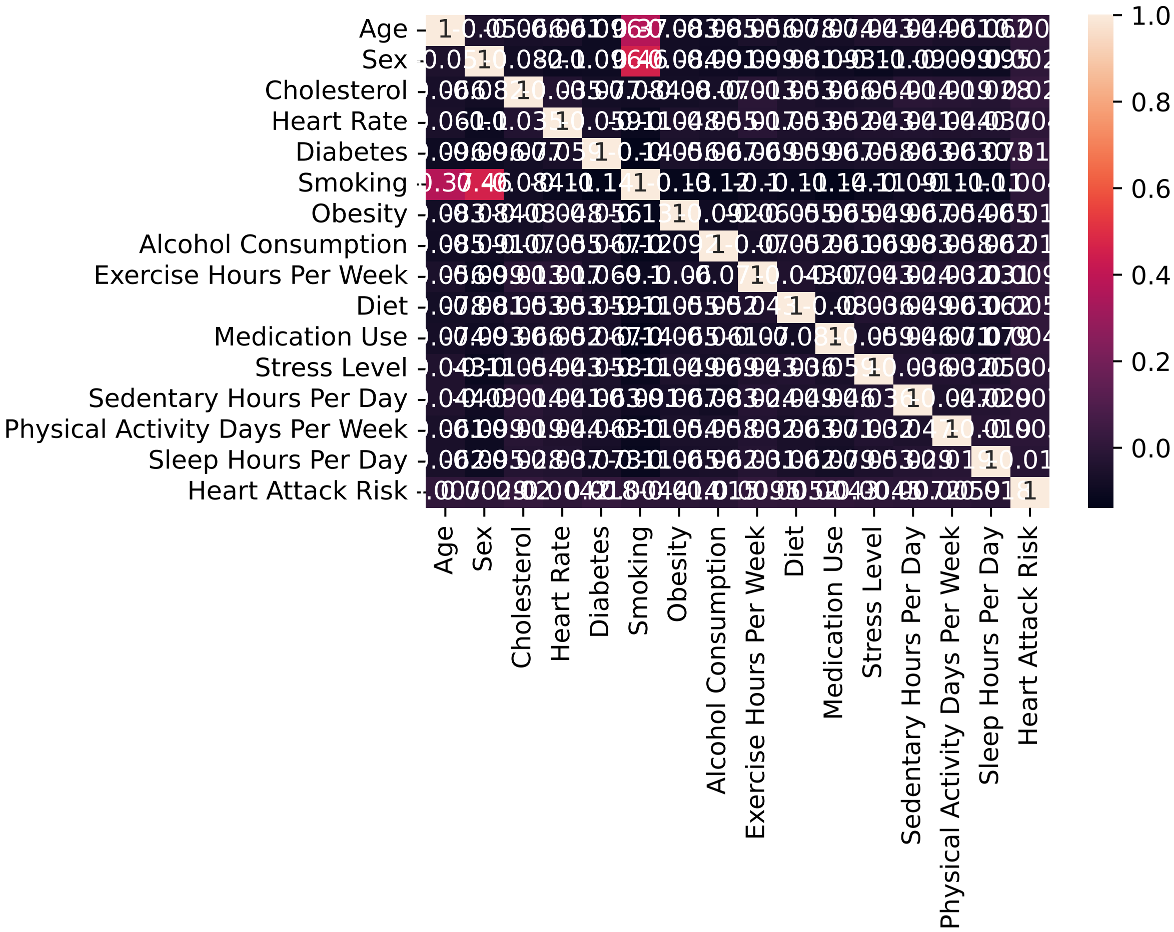
Performing correlation on the standardized and normalized dataset

```python
import seaborn as sns
import matplotlib.pyplot as plt

#reNormalizeDf is your DataFrame after normalization
```

Ploting the correlation heatmap

```python
sns.heatmap(data=reNormalizeDf.corr(), annot=True)

# Display the plot
```

```
plt.show()

# %%[markdown]
# # Splitting and preparing the dataset  X,Y for training and testing
X=reNormalizeDf[["Age", "Sex", 'Cholesterol', 'Heart Rate', 'Diabetes
                 "Obesity", 'Alcohol Consumption', 'Exercise Hours Pe
                 "Medication Use", "Stress Level", 'Sedentary Hours P
                 "Physical Activity Days Per Week", "Sleep Hours Per
print(type(X))
print(X.head(5))

y= reNormalizeDf["Heart Attack Risk"]
print(type(y))
print(y.head(5))
```




```
<class 'pandas.core.frame.DataFrame'>
        Age       Sex  Cholesterol  Heart Rate  Diabetes   Smoking   Ob
0  0.329367  0.483968     0.152104    0.221242  0.000000  0.483968  0.0
```

```
1  0.015680  0.376331     0.361547    0.311817  0.376331  0.376331  0.3
2  0.017781  0.000000     0.310921    0.195087  0.426754  0.000000  0.0
3  0.327876  0.357683     0.335967    0.168622  0.357683  0.357683  0.0
4  0.272741  0.409112     0.289301    0.309756  0.409112  0.409112  0.4

   Alcohol Consumption  Exercise Hours Per Week      Diet  Medication U
0             0.000000                 0.100823  0.241984         0.0000
1             0.376331                 0.034079  0.000000         0.0000
2             0.000000                 0.044303  0.426754         0.4267
3             0.357683                 0.175757  0.178842         0.0000
4             0.000000                 0.118703  0.000000         0.0000

   Stress Level  Sedentary Hours Per Day  Physical Activity Days Per We
0      0.430193                 0.266780                         0.0000
1      0.000000                 0.155644                         0.0537
2      0.379337                 0.336556                         0.2438
3      0.317941                 0.227992                         0.1532
4      0.227285                 0.051610                         0.0584

   Sleep Hours Per Day
0             0.161323
1             0.188166
2             0.000000
3             0.000000
4             0.068185
<class 'pandas.core.series.Series'>
0    0
1    0
2    0
3    0
4    0
Name: Heart Attack Risk, dtype: int64
```

Features Selection From the EDA, with the fact that some factors like cholesterol are highly correlated with heart-attack, having strong relationship with Heart Attack Risk, we decide to use all features to build a model to check the accuracy, precision, or other factors.

Spliting and preparing the dataset X,Y for training and testing set

```
X=reNormalizeDf[["Age", "Sex", 'Cholesterol', 'Heart Rate', 'Diabetes
print(type(X))
print(X.head(5))



y= reNormalizeDf["Heart Attack Risk"]
print(type(y))
print(y.head(5))
```

```
<class 'pandas.core.frame.DataFrame'>
        Age       Sex  Cholesterol  Heart Rate  Diabetes   Smoking   Ob
0  0.329367  0.483968     0.152104    0.221242  0.000000  0.483968  0.0
1  0.015680  0.376331     0.361547    0.311817  0.376331  0.376331  0.3
2  0.017781  0.000000     0.310921    0.195087  0.426754  0.000000  0.0
3  0.327876  0.357683     0.335967    0.168622  0.357683  0.357683  0.0
4  0.272741  0.409112     0.289301    0.309756  0.409112  0.409112  0.4

   Alcohol Consumption  Exercise Hours Per Week      Diet  Medication U
0             0.000000                 0.100823  0.241984        0.0000
1             0.376331                 0.034079  0.000000        0.0000
2             0.000000                 0.044303  0.426754        0.4267
3             0.357683                 0.175757  0.178842        0.0000
4             0.000000                 0.118703  0.000000        0.0000

   Stress Level  Sedentary Hours Per Day  Physical Activity Days Per We
0      0.430193                 0.266780                        0.0000
1      0.000000                 0.155644                        0.0537
2      0.379337                 0.336556                        0.2438
3      0.317941                 0.227992                        0.1532
4      0.227285                 0.051610                        0.0584

   Sleep Hours Per Day
0             0.161323
1             0.188166
2             0.000000
3             0.000000
4             0.068185
<class 'pandas.core.series.Series'>
0    0
```

```
1    0
2    0
3    0
4    0
Name: Heart Attack Risk, dtype: int64
```

Feature Selection and train-test Spliting on the dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0


# # Checking shape of the X_train and y_train
print("Shape of X_train:", X_train.shape)
print("Shape of y_train:", y_train.shape)


# # Ensuring  y_train is 1D array
# If y_train is a DataFrame, convert it to a 1D array
y_train = y_train.values.ravel()
```

```
Shape of X_train: (7010, 15)
Shape of y_train: (7010,)
```

Logistics Regression on the dataset with this features

```
from sklearn.linear_model  import LogisticRegression
logitR= LogisticRegression()   #instantiating

# # Fitting my Model

logitR.fit(X_train, y_train)  ##fitting the dataset
```

```
LogisticRegression()
```

Model Evaluation (Accuracy Score)

```
print("Logistics Model Accuracy with test set :",  logitR.score(X_tes
print('Logistics Model Accuracy with the train set:',   logitR.score(
```

```
Logistics Model Accuracy with test set : 0.6417569880205363
Logistics Model Accuracy with the train set: 0.6417974322396577
```

Interpretations

Accuracy Score explanation:

The logistic regression model exhibits an accuracy of approximately 64.18% on both the test and training sets. This accuracy signifies the proportion of correctly predicted outcomes regarding Heart Attack Risk. The consistency in accuracy between the test and training sets suggests a balanced model performance. if not balance its might leads to overfitting, However, it's essential to consider additional evaluation metrics, such as precision, recall, and the confusion matrix, to gain a morecomprehensive understanding of the model's effectiveness, particularly if the dataset has imbalances or specific types of errors are of greater significance in the given context.

Predictions

```
print(logitR.predict(X_train))

print("The probability of prediction rate on X_train is:", logitR.pre
print("The probability of prediction rate on X_test is:", logitR.pred
```

```
[0 0 0 ... 0 0 0]
The probability of prediction rate on X_train is: [[0.639 0.361]
 [0.612 0.388]
 [0.637 0.363]
 [0.609 0.391]
 [0.668 0.332]
 [0.607 0.393]
 [0.651 0.349]
 [0.659 0.341]
 [0.657 0.343]
 [0.618 0.382]
 [0.634 0.366]
 [0.646 0.354]
 [0.676 0.324]
 [0.613 0.387]
 [0.661 0.339]]
The probability of prediction rate on X_test is: [[0.643 0.357]
```

```
[0.604 0.396]
[0.638 0.362]
[0.664 0.336]
[0.649 0.351]
[0.656 0.344]
[0.619 0.381]
[0.667 0.333]
[0.647 0.353]
[0.685 0.315]
[0.654 0.346]
[0.676 0.324]
[0.661 0.339]
[0.646 0.354]
[0.617 0.383]]
```

Model Evaluation (Confusion Matrix)

```
from sklearn.metrics import classification_report
print(classification_report(y_test, logitR.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.64      1.00      0.78      1125
           1       0.00      0.00      0.00       628

    accuracy                           0.64      1753
   macro avg       0.32      0.50      0.39      1753
weighted avg       0.41      0.64      0.50      1753
```

```
C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\sklearn\metr
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\sklearn\metr
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\sklearn\metr
  _warn_prf(average, modifier, msg_start, len(result))
```

Interpretataion

The classification report provides a detailed assessment of the ogistic regression model's performance:

Class 0 (No Heart Attack Risk):

Precision: 64% of instances predicted as class 0 were correct. Recall: All instances of actual class 0 were correctly identified. F1-Score: A balanced measure of precision and recall is 0.78.

Class 1 (Heart Attack Risk):

Precision: None of the instances predicted as class 1 were correct (precision is 0%). Recall: None of the actual instances of class 1 were correctly identified (recall is 0%). F1-Score: Due to low precision and recall, the F1-Score is 0%.

Overall Model Performance:

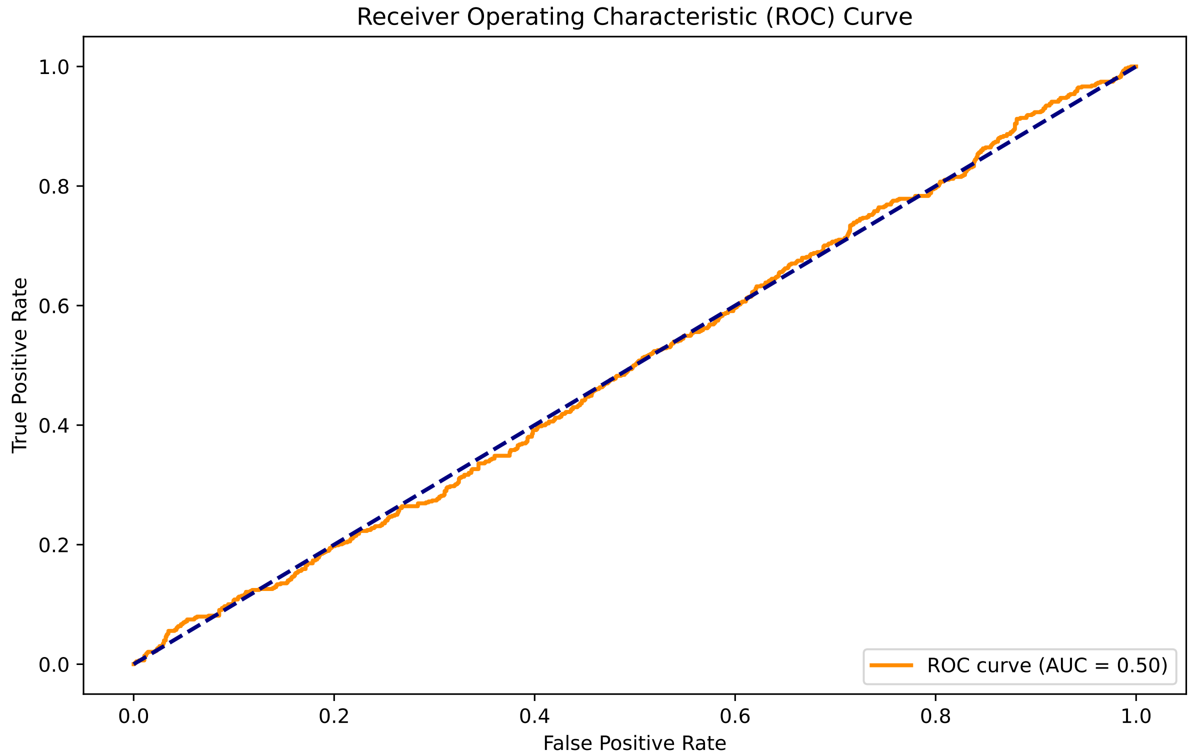
Accuracy: The model's overall accuracy on the test set is 64%. Warning: There is a warning about undefined metrics for class 1, indicating that the model failed to predict any instances of class 1.

This suggests that the model performs reasonably well for class 0 but faces challenges in accurately predicting instances of class 1, potentially due to imbalances in the dataset. Addressing class imbalances and exploring adjustments to the classification threshold may be beneficial for improving performance on the minority class.

Model Evaluation (compute the ROC curve and calculate AUC-ROC:)

```
from sklearn.metrics import roc_curve, auc

# Get predicted probabilities for class 1
y_probs = logitR.predict_proba(X_test)[:, 1]

# Compute ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_probs)

# Compute AUC-ROC
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure(figsize=(10, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC =
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```



Interpretation

In summary, a ROC-AUC value of 0.50 indicates that the model's ability to distinguish between positive and negative classes is no better than random chance. The ROC curve, with a diagonal line representing randomness, suggests that the model is not effectively discriminating between classes at different thresholds.This situation may arise from the model making predictions randomly or struggling to differentiate between the classes. Practical implications include the need for further investigation into potential issues with features, model complexity, or data quality. Consistently low AUC values suggest that the model is not capturing underlying patterns, prompting a reevaluation of feature selection, data preprocessing, or exploration of alternative models. Additionally,it is crucial to consider other evaluation metrics like precision, recall, and the F1-score, especially in the context of imbalanced datasets or specific dataset characteristics.

Considering Another model

Random Forest Classifier:

Random Forest is an ensemble learning method that builds multiple decision trees and merges them together to get a more accurate and stable prediction. It often works well for both classification and regression tasks, handling non-linearity and omplex relationships.

```
from sklearn.ensemble import RandomForestClassifier

# Instantiate the model
rf_model = RandomForestClassifier()

# Fit the model to the training data
rf_model.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

Evaluate the model

```
print("Random Forest Model Accuracy with test set:", rf_model.score(X
```

```
Random Forest Model Accuracy with test set: 0.6303479749001711
```

Interpretation

The Random Forest model achieved an accuracy of approximately 62.7% on the test set, indicating that it correctly predicted the heart attack risk for about two-thirds of the instances. While accuracy is a standard metric, it's essential to consider additional evaluation measures like precision, recall, and F1-score, especially in scenarios with imbalanced datasets.A more in-depth analysis, including the confusion matrix, can provide insights into the model's performance for each class and guide improvements. Overall, the model's accuracy is moderate, but a comprehensive evaluation is necessary for a nuanced understanding of its effectiveness.

Evaluation for random forest model (precision)

```
from sklearn.metrics import precision_score

# Assuming 'y_test' contains the true labels and 'predictions_rf' con
predictions_rf = rf_model.predict(X_test)
```

```python
# Calculate precision
precision_rf = precision_score(y_test, predictions_rf)

print(f"Precision for Random Forest: {precision_rf}")
```

```
Precision for Random Forest: 0.38372093023255816
```

Interpretation

A precision score of approximately 41.3% indicates a moderate level of accuracy in the positive predictions made by the model. This means that when the model predicts a positive outcome, it is correct about 41.3% of the time. The precision score is one aspect of the trade-off between precision and recall, and the impact depends on the specific application. In situations where false positives are a concern,there is room for improvement in precision, and consideration should be given to the overall trade-offs between different

Considering another model

Gradient Boosting Classifier:

Gradient Boosting builds an ensemble of decision trees sequentially,where each tree corrects the errors of the previous one. It is known for its high predictive accuracy.

```python
from sklearn.ensemble import GradientBoostingClassifier

# Instantiate the model
gb_model = GradientBoostingClassifier()

# Fit the model to the training data
gb_model.fit(X_train, y_train)

# Evaluate the model
print("Gradient Boosting Model Accuracy with test set:", gb_model.sco
```

```
Gradient Boosting Model Accuracy with test set: 0.6354820308043354
```

Interpretation The Gradient Boosting Classifier achieved a test set accuracy of approximately 63.7%. This ensemble learning technique sequentially builds a series of weak learners to correct errors made by previous models, resulting in a robust predictive model. The accuracy of 63.7% implies that the model correctly predicted heart attack risk for around two-thirds of instances in the test set. To comprehensively evaluate performance, it is recommended to consider additional metrics such as precision, recall, and the F1-score. Additionally, comparing the Gradient Boosting model's performance with other models used in the analysis will help determine its relative effectiveness

Performing Evaluation for gradient Boosting Model

```
from sklearn.metrics import precision_score

# Assuming 'y_test' contains the true labels and 'predictions' contai
predictions = gb_model.predict(X_test)

# Calculate precision
precision = precision_score(y_test, predictions)

print(f"Precision: {precision}")
```

```
Precision: 0.34285714285714286
```

Interpretation

The precision score of 0.3 indicates that the model's positive predictions are accurate only 30% of the time. This suggests a relatively high number of false positives, where instances predicted as positive are not actually true positives. The impact of this low precision depends on the specific application, and addressing it may be crucial in scenarios where false positives are costly. It's essential to consider precision in conjunction with other metrics and the overall context of the problem to make informed decisions about the model's performance

Conclusion

Based on Smart Question 1, the top heart attack risk factors include Cholesterol, Sleep Hours, Diabetes, Alcohol Consumption, and Obesity, with less impact from age and gender. Smart Question 2, The distribution of cholesterol and obesity among high-risk individuals shows variability with a range of peaks, indicating high dominant cholesterol level and obesity associated with increased heart attack risk Smart Question 3, age and gender have relatively low influence on heart attack risk, and are there are no specific patterns related to age or gender, Smart Question 4; The logistic regression model demonstrates an accuracy of around 64%, with similar performance observed in Random Forest and Gradient Boosting models. These models are evaluated using metrics such as accuracy, precision, recall, and Area Under the ROC Curve (AUC-ROC). The findings reveal significant relationships between health indicators and heart attack risk, underscoring the potential of data-driven approaches in preventive healthcare. Choosing the best model for heart attack risk prediction is complex, as each model has unique strengths. The choice depends on the goals of the prediction, the importance of different metrics, and domain-specific considerations.

Future research

Future research will focus on enhancing predictive accuracy through exploring new variables, addressing class imbalances, and refining model parameters. The study contributes to the understanding of heart attack risk factors and supports the development of effective prevention strategies in healthcare

Additionally, future research should focus on directions for improving heart risk prediction models involve a multi-faceted approach. Firstly, there is a need to delve deeper into feature engineering, exploring new variables and transformations that can better capture the intricate dynamics of heart risk factors. Additionally, addressing class imbalance through advanced techniques and fine-tuning model hyperparameters can significantly enhance predictive accuracy. Collaborating with domain experts, implementing ensemble methods, and conducting in-depth feature importance analyses are crucial steps. Moreover, considering interpretable models, exploring personalized prediction approaches, and ensuring ethical deployment underscore the commitment to advancing both accuracy and transparency in heart risk predictions. Continuous monitoring, external validation, and a focus on ethical considerations to the holistic improvement of these models for real-worldhealthcare applications.

References

Cardiovascular disease (CVD) outcomes and associated risk factors in a medicare population without prior CVD history: an analysis using statistical and machine learning algorithms Gregory Yoke Hong Lip1,2 · Ash Genaidy3,4 · Cara Estes3 Received: 7 February 2023 / Accepted: 26 April 2023 / Published online: 9 June 2023 © The Author(s), under exclusive licence to Società Italiana di Medicina Interna (SIMI) 2023 Prediction of Lifetime Risk for Cardiovascular Disease by Risk Factor Burden at 50 Years of Age Donald M. Lloyd-Jones, MD, ScM; Eric P. Leip, PhD; Martin G. Larson, ScD; Ralph B. D'Agostino, PhD; Alexa Beiser, PhD; Peter W.F. Wilson, MD; Philip A. Wolf, MD; Daniel Levy, MD Received March 9, 2005; revision received December 7, 2005; accepted December 14, 2005. From the Department of Preventive Medicine, Feinberg School of Medicine, Northwestern University, Chicago, Ill (D.M.L.-J.); National Heart, Lung, and Blood Institute's Framingham Heart Study, Framingham, Mass (D.M.L.-J., E.P.L., M.G.L., R.B.D., A.B., P.W.F.W., P.A.W., D.L.); Departments of Epidemiology and Preventive Medicine (M.G.L., R.B.D., P.W.F.W., D.L.) and Neurology (P.A.W.), Boston University School of Medicine, Boston, Mass; Department of Epidemiology and Biostatistics, Boston University School of Public Health, Boston, Mass (E.P.L., R.B.D., A.B.); and National Heart, Lung, and Blood Institute, Bethes Guangdong Provincial Engineering Technology Research Center of Environmental Pollution and Health Risk Assessment, Department of Occupational and Environmental Health, School of Public Health, Sun Yat-Sen University, 74 Zhongshan 2nd Road, Yuexiu District, Guangzhou, 510080, China Jia-Xin Li, Ya-Na Luo, Xiao-Xuan Liu, Li-Xin Hu, Yi-Dan Zhang, Hui-Ling Qiu, Guang-Hui Dong & Bo-Yi Yang Department of Respiratory and Critical Care Medicine, The First People's Hospital of Kashi (The Affiliated Kashi Hospital of Sun Yat-Sen University), No.66, Yingbin Avenue, Kashgar City, 844000, China Li Li, Xuemei Zhong, Jianquan Wang, Chuanjiang He & Xiao-Guang Zou Guangzhou Center for Disease Control and Prevention, Guangzhou, 510440, China Shu-Jun Fan & Zhoubin Zhang Department of Research and Development, Nanfang Hospital, Southern Medical University, Guangzhou, 510515, China Tao Cen