Anderson Germano de Souza - RA: 12326462

Conjunto de Dados de Previsão de Risco de Ataque Cardíaco

Desbloqueando Insights Preditivos com Conjunto de Dados de Ataque Cardíaco Sintético Multifacetado.

Sobre o conjunto de dados

Contexto

O Conjunto de Dados de Previsão de Risco de Ataque Cardíaco serve como um recurso valioso para se aprofundar na intrincada dinâmica da saúde cardíaca e seus preditores. Ataques cardíacos, ou infartos do miocárdio, continuam a ser um problema de saúde global significativo, exigindo uma compreensão mais profunda de seus precursores e potenciais fatores mitigadores. Este conjunto de dados encapsula uma gama diversificada de atributos, incluindo idade, níveis de colesterol, pressão arterial, hábitos de fumar, padrões de exercício, preferências alimentares e muito mais, com o objetivo de elucidar a complexa interação dessas variáveis na determinação da probabilidade de um ataque cardíaco. Ao empregar análise preditiva e aprendizado de máquina neste conjunto de dados, pesquisadores e profissionais de saúde podem trabalhar em direção a estratégias proativas para prevenção e gerenciamento de doenças cardíacas. O conjunto de dados é uma prova dos esforços coletivos para melhorar nossa compreensão da saúde cardiovascular e pavimentar o caminho para um futuro mais saudável.

Conteúdo

Este conjunto de dados fornece uma gama abrangente de recursos relevantes para a saúde do coração e as escolhas de estilo de vida, abrangendo detalhes específicos do paciente, como idade, sexo, níveis de colesterol, pressão arterial, frequência cardíaca e indicadores como diabetes, histórico familiar, hábitos de fumar, obesidade e consumo de álcool. Além disso, fatores de estilo de vida como horas de exercício, hábitos alimentares, níveis de estresse e horas sedentárias estão incluídos. Aspectos médicos que incluem problemas cardíacos anteriores, uso de medicamentos e níveis de triglicerídeos são considerados. Aspectos socioeconômicos, como renda e atributos geográficos, como país, continente e hemisfério, são incorporados. O conjunto de dados, que consiste em 8763 registros de pacientes de todo o mundo, culmina em um recurso crucial de classificação binária que denota a presença ou ausência de um risco de ataque cardíaco, fornecendo um recurso abrangente para análise preditiva e pesquisa em saúde cardiovascular.

Glossário do Conjunto de Dados (em termos de coluna)

Patient ID - Identificador exclusivo para cada paciente
Age - Idade do paciente
Sex - Sexo do paciente (0: Masculino / 1: Feminino)
Cholesterol - Níveis de colesterol do paciente
Heart Rate - Frequência cardíaca do paciente
Diabetes - Se o paciente tem diabetes (Sim/Não)
Family History - História da família de problemas relacionados ao coração (1: Sim, 0: Não)
Smoking - Status de fumar do paciente (1: Fumante, 0: Não fumante)
Obesity - Estado de obesidade do paciente (1: Obesa, 0: Não obesa)
Alcohol Consumption - Nível de consumo de álcool pelo paciente (Nenhum/Leve/Moderado/Pesado)
Exercise Hours Per Week - Número de horas de exercício por semana
Diet - Hábitos alimentares do paciente (0: Saudável / 1: Média / 2: Insalubre)
Previous Heart Problems - Problemas Cardíacos Anteriores do paciente (1: Sim, 0: Não)
Medication Use - Uso de Medicamentos pelo paciente (1: Sim, 0: Não)
Stress Level - Nível de Estresse relatado pelo paciente (1-10)
Sedentary Hours Per Day - Horas de atividade sedentária por dia
Income - Nível de renda do paciente
BMI - Índice de Massa Corporal (IMC) do paciente
Triglycerides - Níveis de triglicerídeos do paciente
Physical Activity Days Per Week - Dias de atividade física por semana
Sleep Hours Per Day - Horas de sono por dia
Heart Attack Risk - Presença de risco de ataque cardíaco (1: Sim, 0: Não)

Análise Detalhada do Código de Predição de Ataque Cardíaco:

Este código utiliza Python e bibliotecas de aprendizado de máquina para analisar dados médicos e criar um modelo capaz de prever o risco de um paciente sofrer um ataque cardíaco. O código se divide em etapas principais:

1. Coleta e Preparo dos Dados:

- Os dados, provenientes de arquivos CSV chamados "train.csv", "test.csv" e "y_test.csv", são carregados utilizando a biblioteca Pandas.
- A coluna "PatientID", irrelevante para a predição, é removida.
- A qualidade dos dados é verificada, buscando por valores faltantes. Se encontrados, estes são preenchidos com a mediana dos valores da respectiva coluna.
- As variáveis categóricas, como sexo e histórico de tabagismo, são convertidas para representações numéricas através do One-Hot Encoding, preparando-as para o algoritmo de aprendizado de máquina.

2. Investigando Relações nos Dados:

- Gráficos como histogramas e scatter plots são gerados com o Matplotlib e Seaborn para visualizar a distribuição dos dados e identificar possíveis relações entre as variáveis.
- A correlação entre as variáveis numéricas é analisada através de um mapa de calor (heatmap).

3. Construindo e Treinando o Modelo Preditivo:

- O código utiliza um algoritmo de Árvore de Decisão para construir o modelo preditivo.
- Os dados são divididos em conjuntos de treinamento e validação para avaliar a capacidade de generalização do modelo.
- O modelo é treinado com os dados de treinamento, aprendendo a classificar o risco de ataque cardíaco com base nos dados médicos e de estilo de vida.

4. Avaliando a Performance do Modelo:

- O modelo treinado é utilizado para fazer previsões nos dados de validação.
- A acurácia, que mede a porcentagem de previsões corretas, é calculada para determinar a performance do modelo.

5. Prevendo o Risco de um Novo Paciente:

- Uma função interativa, prever_risco(), permite a entrada de dados de um novo paciente, como idade, colesterol, histórico familiar, etc.
- A função processa esses dados, aplicando as mesmas transformações realizadas nos dados de treinamento, e os utiliza no modelo

para gerar a predição de risco de ataque cardíaco.

Em resumo, o código demonstra um processo completo de análise de dados e aprendizado de máquina para a predição de riscos à saúde, desde a manipulação e exploração dos dados até a construção e aplicação de um modelo preditivo.

1. Importação das Bibliotecas

Importa bibliotecas essenciais para análise de dados (pandas, numpy), visualização (matplotlib, seaborn) e aprendizado de máquina (scikit-learn).

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split # Para validação cruzada
from sklearn.preprocessing import OneHotEncoder # Para codificação One-Hot
```

1. Carregamento e Preparação dos Dados:

- Carrega dados de treinamento e teste a partir de arquivos CSV.
- Remove a coluna 'PatientID', que provavelmente não é relevante para a previsão.
- Realiza uma análise exploratória de dados (EDA) para entender melhor os dados.

In [2]:
```
# Carregando os dados
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
y_test = pd.read_csv('y_test.csv')
```

In [3]:
```
# Removendo a coluna 'PatientID'
train.drop(['PatientID'], axis=1, inplace=True)
test.drop(['PatientID'], axis=1, inplace=True)
```

In [4]:
```
train.dtypes
```

Out[4]:
```
Age                            int64
Sex                            int64
Cholesterol                    int64
HeartRate                      int64
Diabetes                       int64
FamilyHistory                  int64
Smoking                        int64
Obesity                        int64
AlcoholConsumption             int64
ExerciseHoursPerWeek         float64
Diet                           int64
PreviousHeartProblems          int64
MedicationUse                  int64
StressLevel                    int64
SedentaryHoursPerDay         float64
BMI                            int64
Triglycerides                  int64
PhysicalActivityDaysPerWeek    int64
SleepHoursPerDay               int64
HeartAttackRisk                int64
dtype: object
```

In [5]:
```
train.head()
```

Out[5]:

	Age	Sex	Cholesterol	HeartRate	Diabetes	FamilyHistory	Smoking	Obesity	AlcoholConsumption	ExerciseHoursPerWeek	Diet	Previo
0	67	0	208	72	0	0	1	0	0	4.16	1	
1	21	0	389	98	1	1	1	1	1	1.81	2	
2	21	1	324	72	1	0	0	0	0	2.07	0	
3	84	0	383	73	1	1	1	0	1	9.82	1	
4	66	0	318	93	1	1	1	1	0	5.80	2	

In [6]:

```
# Somar o números de dados faltantes por parâmetro
train.isnull().sum().sort_values(ascending=False)
```

Out[6]:

```
Age                          0
Sex                          0
SleepHoursPerDay             0
PhysicalActivityDaysPerWeek  0
Triglycerides                0
BMI                          0
SedentaryHoursPerDay         0
StressLevel                  0
MedicationUse                0
PreviousHeartProblems        0
Diet                         0
ExerciseHoursPerWeek         0
AlcoholConsumption           0
Obesity                      0
Smoking                      0
FamilyHistory                0
Diabetes                     0
HeartRate                    0
Cholesterol                  0
HeartAttackRisk              0
dtype: int64
```

In [7]:

```
train.describe()
```

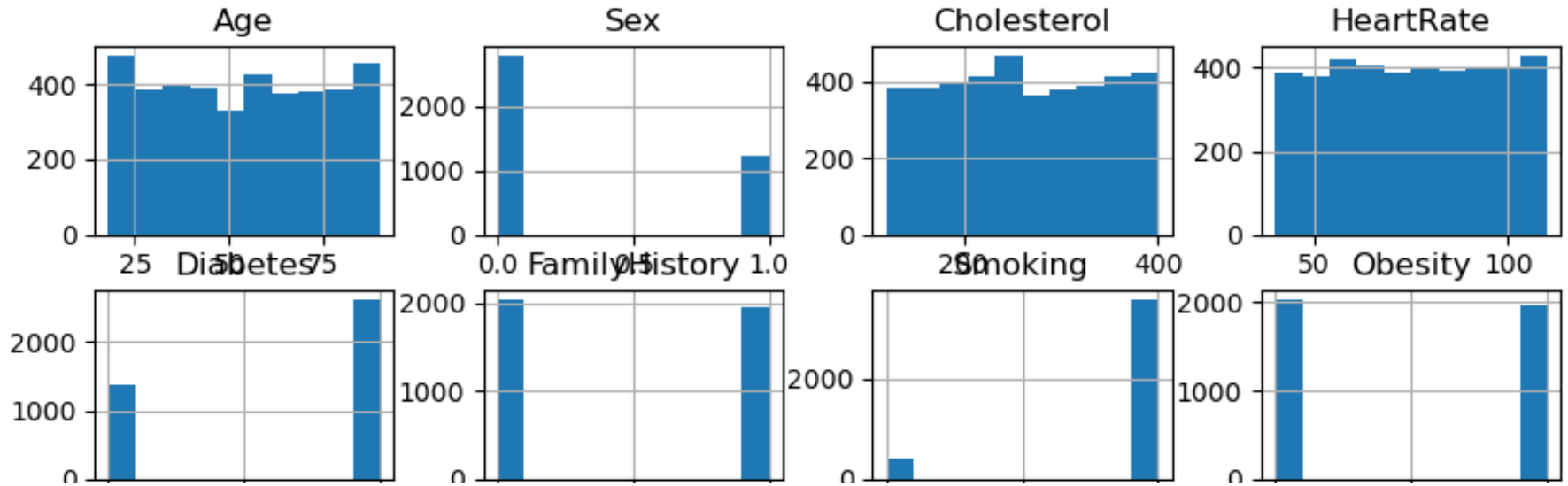
Out[7]:

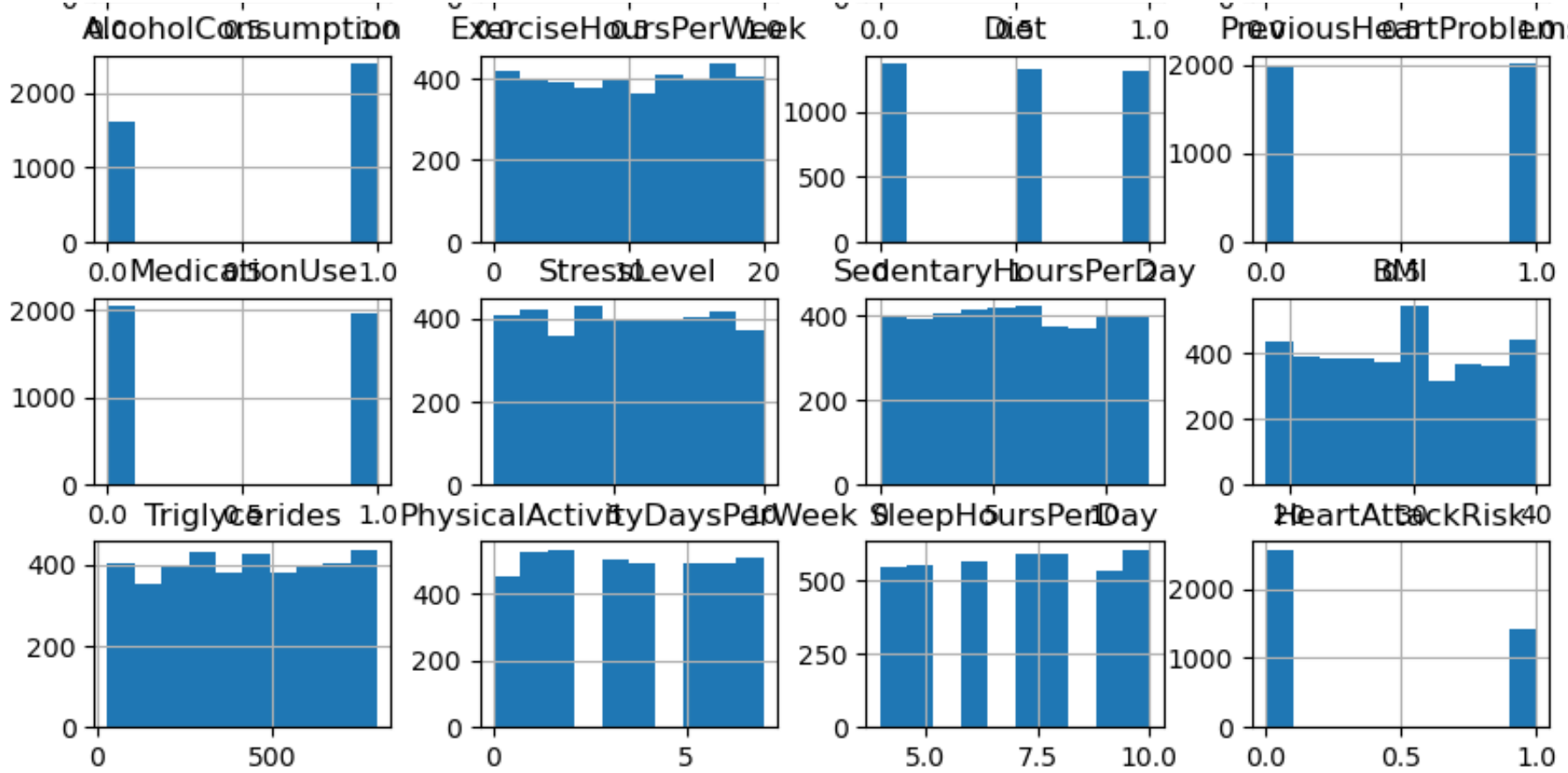
	Age	Sex	Cholesterol	HeartRate	Diabetes	FamilyHistory	Smoking	Obesity	AlcoholConsumptio
count	3999.000000	3999.000000	3999.000000	3999.000000	3999.000000	3999.000000	3999.000000	3999.000000	3999.00000
mean	53.761440	0.303076	260.764441	74.919480	0.654414	0.490873	0.894974	0.493373	0.59789
std	21.506077	0.459645	80.670083	20.366834	0.475619	0.499979	0.306626	0.500019	0.49038
min	18.000000	0.000000	120.000000	40.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	35.000000	0.000000	194.000000	57.000000	0.000000	0.000000	1.000000	0.000000	0.00000
50%	54.000000	0.000000	257.000000	75.000000	1.000000	0.000000	1.000000	0.000000	1.00000
75%	73.000000	1.000000	331.500000	93.000000	1.000000	1.000000	1.000000	1.000000	1.00000
max	90.000000	1.000000	400.000000	110.000000	1.000000	1.000000	1.000000	1.000000	1.00000

1. Visualização dos Dados:

Utiliza bibliotecas de visualização para criar gráficos como histogramas, gráficos de barras, etc., para visualizar padrões nos dados.

In [8]:
```
train.hist(figsize=(10,8));
```





In [9]:

```
# Analisando a distribuição pelo gênero
x = train[['Sex', 'Cholesterol']].groupby(['Sex']).count()
print(x)
```

```
     Cholesterol
Sex
0           2787
1           1212
```

In [10]:

```
# Analisando a distribuição pela idade
x = train[['Age', 'Cholesterol']].groupby(['Age']).count()
print(x)
```

```
     Cholesterol
Age
18            53
19            67
20            51
21            67
22            70
..           ...
86            49
87            56
88            58
89            52
90            72

[73 rows x 1 columns]
```

In [11]:

```
# Analisando a distribuição pela idade e sexo
x = train[['Age', 'Sex', 'Cholesterol']].groupby(['Age','Sex']).count()
print(x)
```

```
         Cholesterol
Age Sex
18  0             37
    1             16
19  0             45
    1             22
20  0             34
...              ...
88  1             17
89  0             35
    1             17
90  0             52
    1             20

[146 rows x 1 columns]
```

In [12]:

```
# Analisando a distribuição pela idade e sexo
x = train[['Age','Sex', 'Diet']].groupby(['Age','Sex']).count()
print(x)
```

```
         Diet
Age Sex
18  0      37
    1      16
19  0      45
    1      22
20  0      34
...       ...
88  1      17
89  0      35
    1      17
90  0      52
    1      20

[146 rows x 1 columns]
```

In [13]:

```
# Analisando a distribuição pela idade e sexo
x = train[['Age','Sex', 'StressLevel']].groupby(['Age','Sex']).count()
print(x)
```

```
         StressLevel
Age Sex
18  0             37
    1             16
19  0             45
    1             22
20  0             34
...              ...
88  1             17
89  0             35
    1             17
90  0             52
    1             20

[146 rows x 1 columns]
```

In [14]:

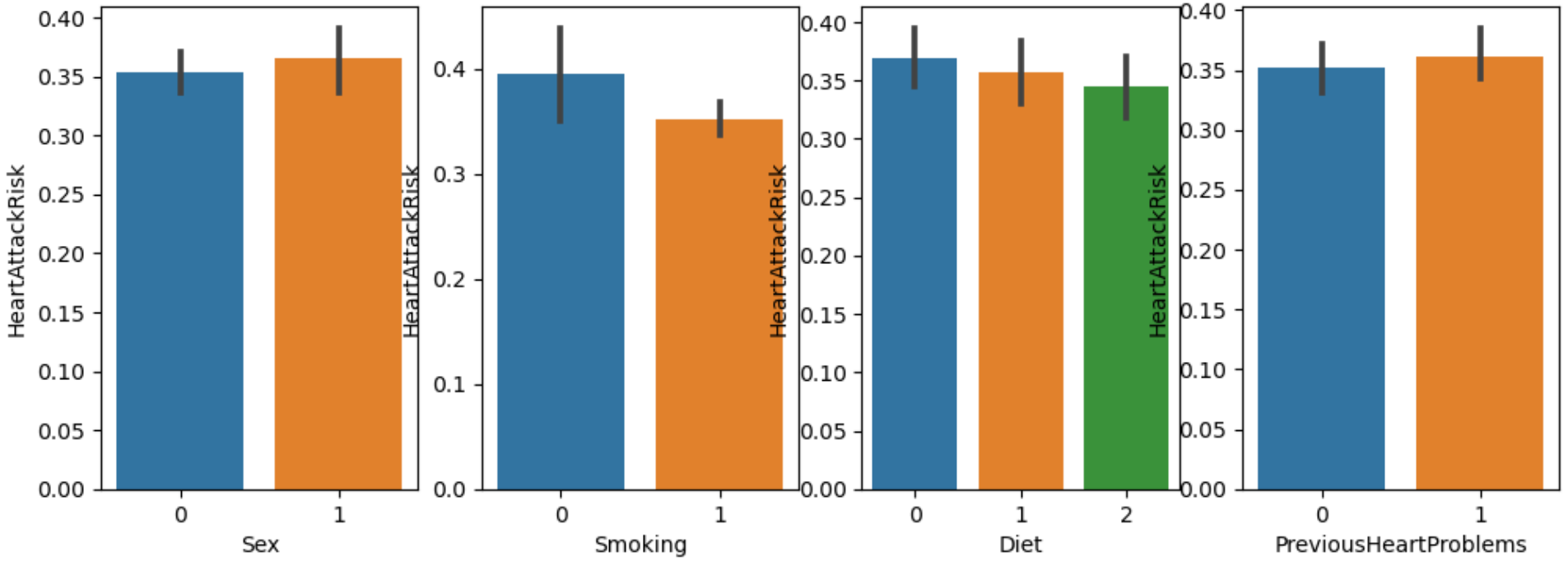
```
# Analisando a chance de risco de infarto pelo gênero
train[['Sex', 'HeartAttackRisk']].groupby(['Sex']).mean()
```

Out[14]:

Sex	HeartAttackRisk
0	0.353427
1	0.365512

In [15]:

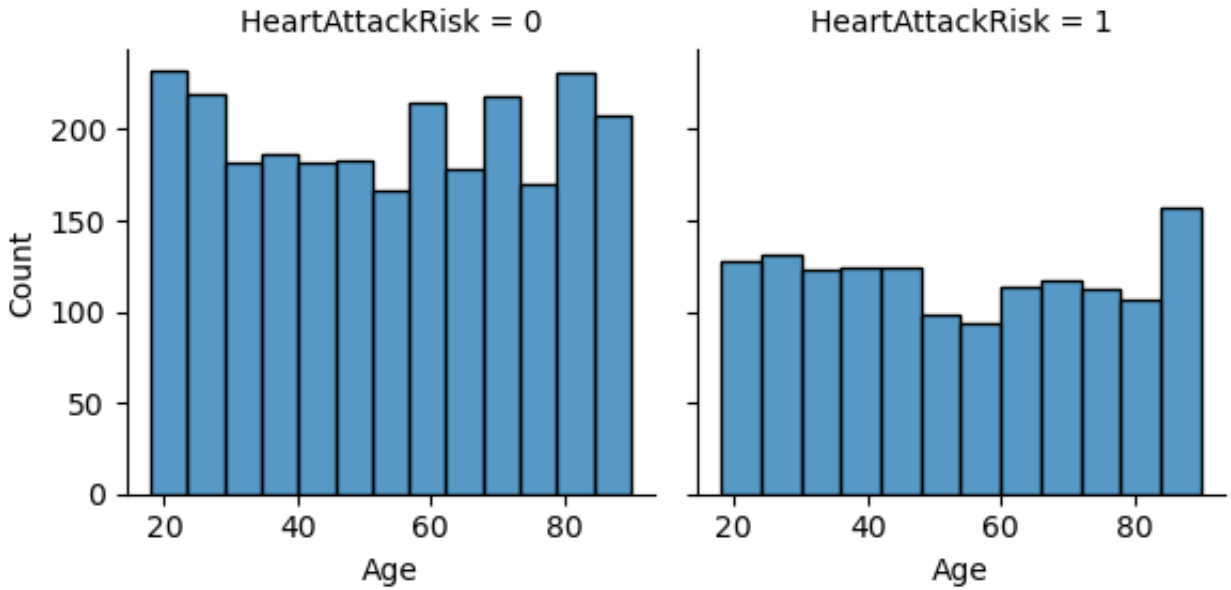
```
# Plotar os gáficos para Heart Attack Risk vs Sex, Smoking, Diet e Previous Heart Problems
fig, (axis1, axis2, axis3, axis4) = plt.subplots(1,4,figsize=(12,4))

sns.barplot(x='Sex', y='HeartAttackRisk', data=train, ax=axis1)
sns.barplot(x='Smoking', y='HeartAttackRisk', data=train, ax=axis2)
sns.barplot(x='Diet', y='HeartAttackRisk', data=train, ax=axis3)
sns.barplot(x='PreviousHeartProblems', y='HeartAttackRisk', data=train, ax=axis4);
```



In [16]:
```
# FaceGrid pelos dados de 'Heart Attack Risk'  - Resultando dois gráficos com risco de ataque cardiaco , mas por i
age_attack = sns.FacetGrid(train, col='HeartAttackRisk')
age_attack.map(sns.histplot, 'Age')
```

```
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x1d6fd647090>
```

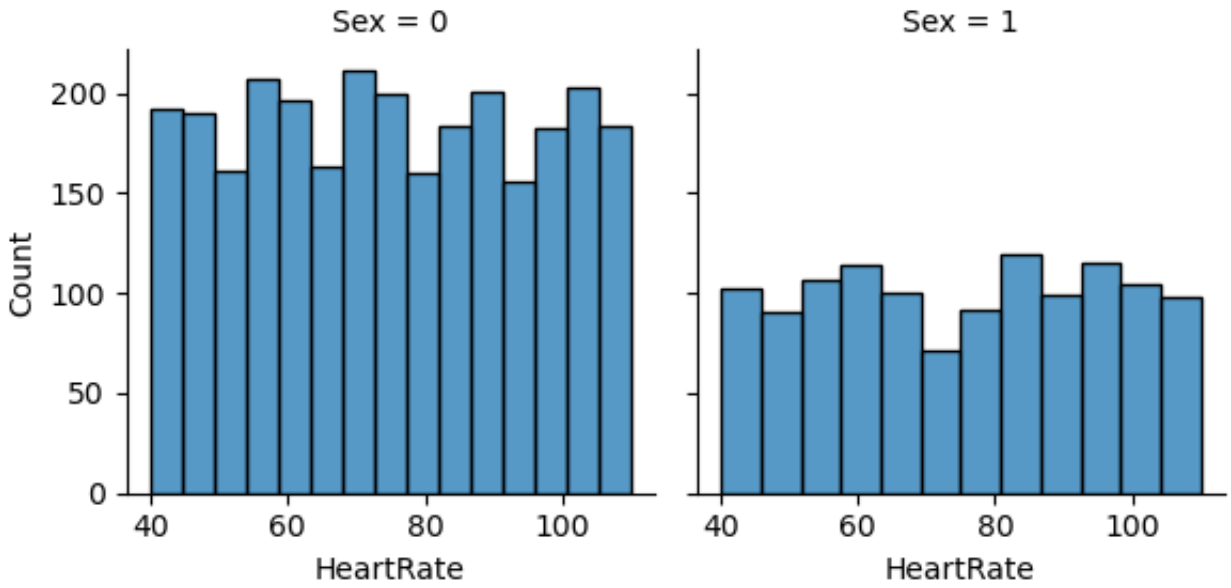


In [17]:
```
# FaceGrid pelos dados de 'Sex' - Resultado dois gráficos com masculino e feminino, mais por 'Heart Rate'
sex_rate = sns.FacetGrid(train, col='Sex')
sex_rate.map(sns.histplot, 'HeartRate')
```

```
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[17]:

```
<seaborn.axisgrid.FacetGrid at 0x1d6fd7b5610>
```

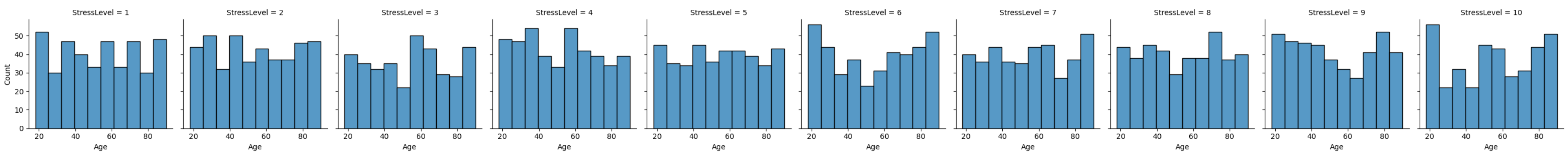


In [18]:

```
# FaceGrid pelos dados de 'Age' - Resultado de 10 gráfico representando os dez levels de Stress pela idade
age_stress = sns.FacetGrid(train, col='StressLevel')
age_stress.map(sns.histplot, 'Age')
```

```
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

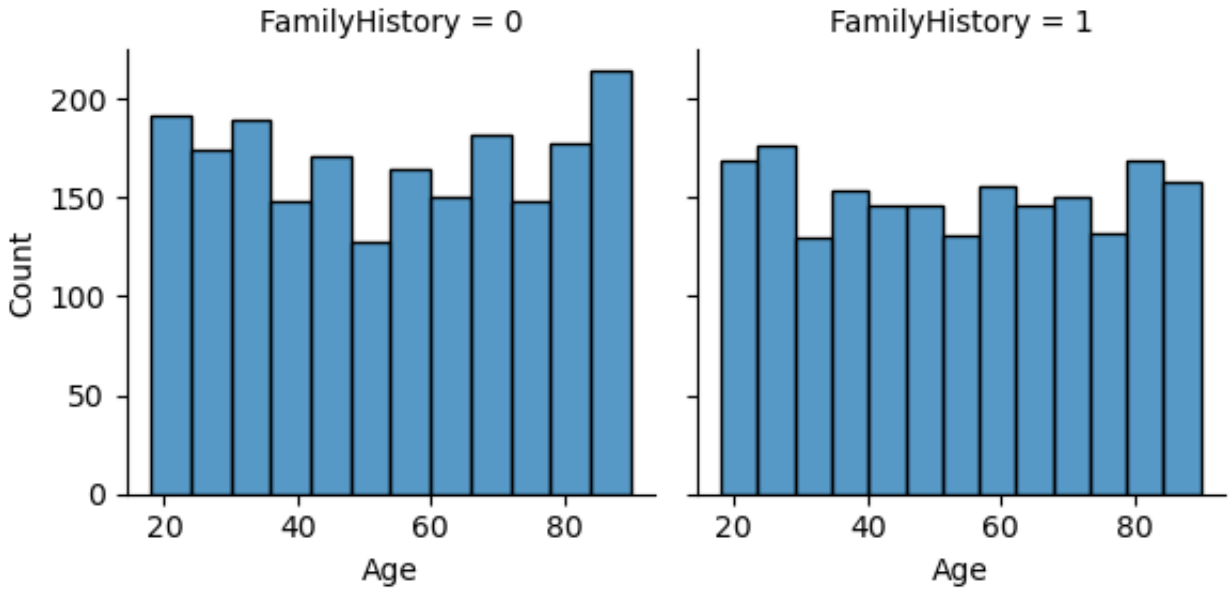
Out[18]:
```
<seaborn.axisgrid.FacetGrid at 0x1d6fd7f0650>
```



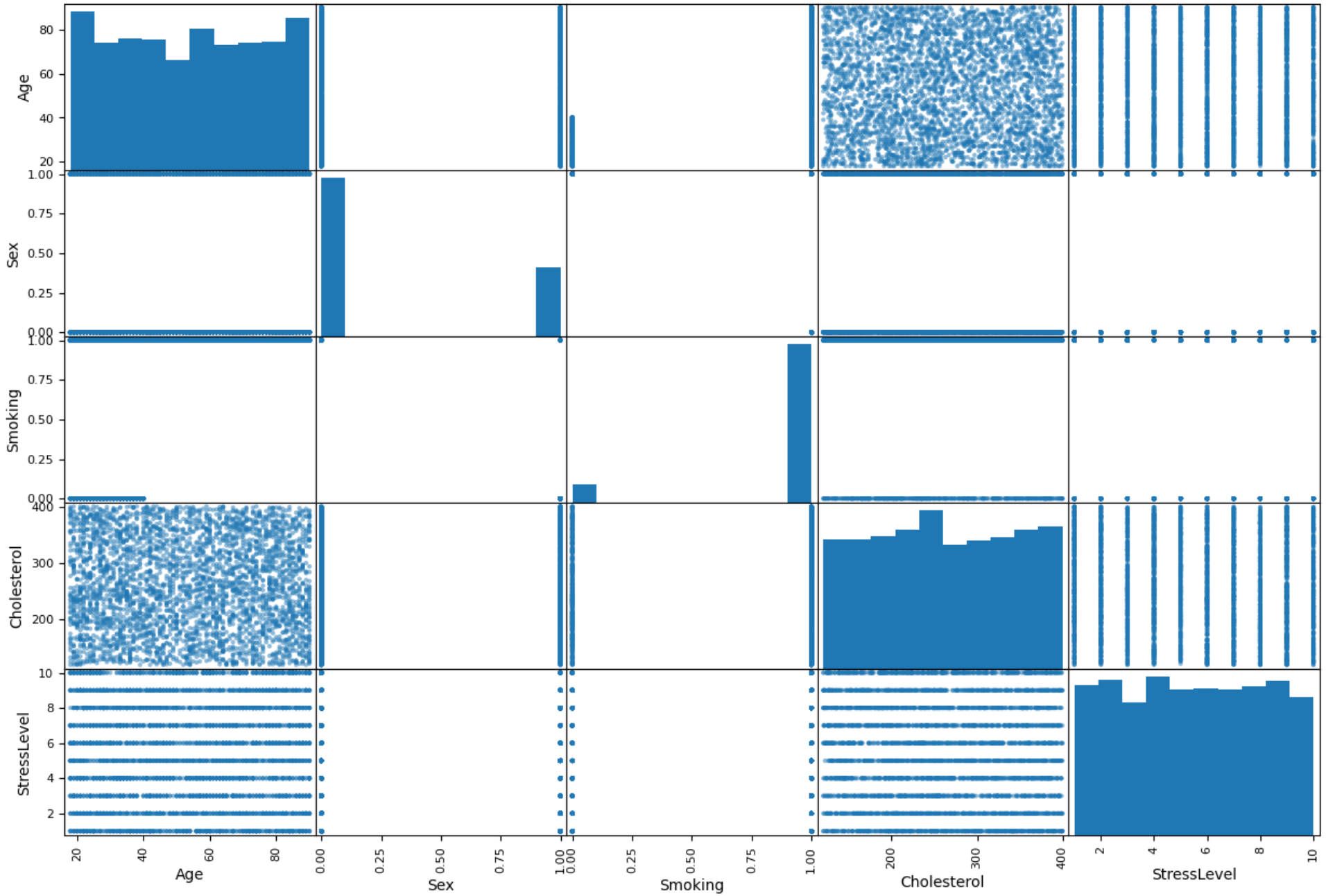
In [19]:
```
# FaceGrid pelos dados de 'Family History'  - Resultando dois gráficos com histórico familiar, mas por idade
age_familyhistory = sns.FacetGrid(train, col='FamilyHistory')
age_familyhistory.map(sns.histplot, 'Age')
```

```
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\sauda\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprec
ated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[19]:
```
<seaborn.axisgrid.FacetGrid at 0x1d6fc7a8d10>
```

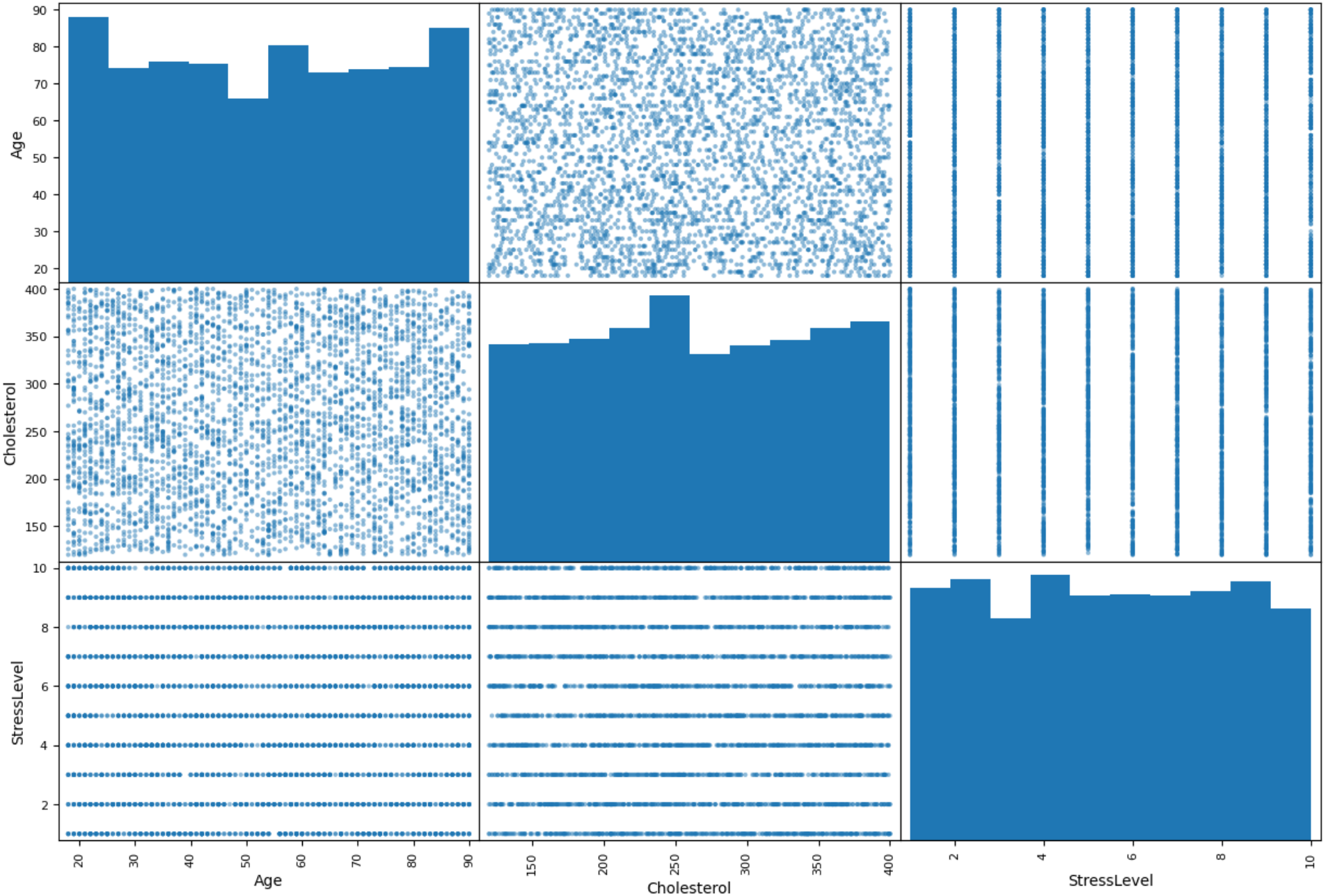


In [20]:
```
# plotar  uma scatter matrix
columns = ['Age', 'Sex', 'Smoking', 'Cholesterol', 'StressLevel']
pd.plotting.scatter_matrix(train[columns], figsize=(15,10));
```


Age
Sex
Smoking
Cholesterol
StressLevel
20
40
60
80
0.00
0.25
0.50
0.75
1.00
200
300
400
2
4
6
8
10

In [21]:

```
# plotar uma scatter matrix
columns = ['Age', 'Cholesterol', 'StressLevel']
pd.plotting.scatter_matrix(train[columns], figsize=(15,10));
```


Age
Cholesterol
StressLevel
Age
Cholesterol
StressLevel

In [22]:

```
# plotar o heatmap para as variaveis numéricas
# Cálculo da correlação
print(train.head())
correlacao = train.corr(numeric_only=True)
print('\nCORRELAÇÃO: \n', correlacao)
# Plot gráfico Heatmap
sns.heatmap(correlacao, cmap='coolwarm', fmt='.2f', linewidths=0.1, vmax=1.0, square=True, linecolor='white', anno
```

```
   Age  Sex  Cholesterol  HeartRate  Diabetes  FamilyHistory  Smoking  \
0   67    0          208         72         0              0        1
1   21    0          389         98         1              1        1
2   21    1          324         72         1              0        0
3   84    0          383         73         1              1        1
4   66    0          318         93         1              1        1

   Obesity  AlcoholConsumption  ExerciseHoursPerWeek  Diet  \
0        0                   0                  4.16     1
1        1                   1                  1.81     2
2        0                   0                  2.07     0
3        0                   1                  9.82     1
4        1                   0                  5.80     2

   PreviousHeartProblems  MedicationUse  StressLevel  SedentaryHoursPerDay  \
0                      0              0            9                  6.61
1                      1              0            1                  4.96
2                      1              1            9                  9.46
3                      1              0            9                  7.64
4                      1              0            6                  1.51

   BMI  Triglycerides  PhysicalActivityDaysPerWeek  SleepHoursPerDay  \
0   31            286                            0                 6
1   27            235                            1                 7
2   28            587                            4                 4
3   36            378                            3                 4
4   22            231                            1                 5

   HeartAttackRisk
0                0
1                0
2                0
```

```
3                 0
4                 0

CORRELAÇÃO:
                                   Age       Sex  Cholesterol  HeartRate  \
Age                           1.000000 -0.022187    -0.008903  -0.005857
Sex                          -0.022187  1.000000    -0.013184   0.011825
Cholesterol                  -0.008903 -0.013184     1.000000   0.001851
HeartRate                    -0.005857  0.011825     0.001851   1.000000
Diabetes                     -0.008991 -0.017333    -0.002233  -0.009845
FamilyHistory                -0.000203 -0.014081    -0.016710  -0.014442
Smoking                       0.395948 -0.519471     0.030842  -0.018537
Obesity                      -0.023570 -0.015202    -0.012422   0.001225
AlcoholConsumption           -0.013580  0.010371    -0.007719  -0.008677
ExerciseHoursPerWeek         -0.020065  0.005423     0.006394  -0.000162
Diet                          0.007576 -0.005342     0.002688   0.027958
PreviousHeartProblems        -0.004267  0.009827     0.015368  -0.028574
MedicationUse                 0.021354  0.007785    -0.013865   0.006467
StressLevel                   0.009038  0.022613    -0.046384  -0.012162
SedentaryHoursPerDay          0.035277 -0.003197     0.002666  -0.005735
BMI                           0.015263  0.007339     0.006640  -0.008719
Triglycerides                 0.005797 -0.019952    -0.021990   0.019355
PhysicalActivityDaysPerWeek -0.008239 -0.002054     0.004431   0.027651
SleepHoursPerDay             -0.004341  0.013754     0.008683  -0.014396
HeartAttackRisk              -0.011901  0.011592     0.012943   0.013301

                              Diabetes  FamilyHistory   Smoking   Obesity  \
Age                          -0.008991      -0.000203  0.395948 -0.023570
Sex                          -0.017333      -0.014081 -0.519471 -0.015202
Cholesterol                  -0.002233      -0.016710  0.030842 -0.012422
HeartRate                    -0.009845      -0.014442 -0.018537  0.001225
Diabetes                      1.000000      -0.021682  0.015185  0.020869
FamilyHistory                -0.021682       1.000000  0.028008  0.006511
Smoking                       0.015185       0.028008  1.000000  0.021562
Obesity                       0.020869       0.006511  0.021562  1.000000
AlcoholConsumption           -0.003971       0.012572  0.010177 -0.026171
ExerciseHoursPerWeek          0.003862       0.005039 -0.005927  0.015957
Diet                          0.006204       0.024159 -0.001877 -0.016744
PreviousHeartProblems         0.010776      -0.005693 -0.010305 -0.006709
MedicationUse                 0.002548      -0.000056 -0.005740 -0.008976
StressLevel                   0.013681       0.020897 -0.022779  0.006992
```

SedentaryHoursPerDay	0.016116	0.008166	0.027058	-0.016499
BMI	0.001189	-0.016199	0.016838	0.009216
Triglycerides	0.014343	-0.004002	0.007147	0.011983
PhysicalActivityDaysPerWeek	0.000773	0.017197	-0.002384	0.019373
SleepHoursPerDay	-0.027086	0.000234	0.019011	-0.019675
HeartAttackRisk	0.018105	-0.007273	-0.027275	-0.006824

	AlcoholConsumption	ExerciseHoursPerWeek	\
Age	-0.013580	-0.020065	
Sex	0.010371	0.005423	
Cholesterol	-0.007719	0.006394	
HeartRate	-0.008677	-0.000162	
Diabetes	-0.003971	0.003862	
FamilyHistory	0.012572	0.005039	
Smoking	0.010177	-0.005927	
Obesity	-0.026171	0.015957	
AlcoholConsumption	1.000000	-0.013229	
ExerciseHoursPerWeek	-0.013229	1.000000	
Diet	-0.016541	-0.012145	
PreviousHeartProblems	0.010826	0.009032	
MedicationUse	0.004621	0.024334	
StressLevel	-0.003342	-0.012457	
SedentaryHoursPerDay	-0.020887	0.013513	
BMI	0.034459	0.014652	
Triglycerides	0.022087	0.013066	
PhysicalActivityDaysPerWeek	0.001501	0.005985	
SleepHoursPerDay	0.007448	-0.012526	
HeartAttackRisk	-0.016818	0.019948	

	Diet	PreviousHeartProblems	MedicationUse	\
Age	0.007576	-0.004267	0.021354	
Sex	-0.005342	0.009827	0.007785	
Cholesterol	0.002688	0.015368	-0.013865	
HeartRate	0.027958	-0.028574	0.006467	
Diabetes	0.006204	0.010776	0.002548	
FamilyHistory	0.024159	-0.005693	-0.000056	
Smoking	-0.001877	-0.010305	-0.005740	
Obesity	-0.016744	-0.006709	-0.008976	
AlcoholConsumption	-0.016541	0.010826	0.004621	
ExerciseHoursPerWeek	-0.012145	0.009032	0.024334	
Diet	1.000000	-0.003615	0.007666	

PreviousHeartProblems	-0.003615	1.000000	-0.004197
MedicationUse	0.007666	-0.004197	1.000000
StressLevel	-0.017945	-0.022240	0.006701
SedentaryHoursPerDay	0.000562	0.005068	0.008403
BMI	-0.013721	-0.000679	0.002361
Triglycerides	0.022980	-0.004783	-0.000727
PhysicalActivityDaysPerWeek	0.014075	-0.009203	-0.022539
SleepHoursPerDay	0.007391	-0.010219	-0.004811
HeartAttackRisk	-0.019790	0.010103	0.005181

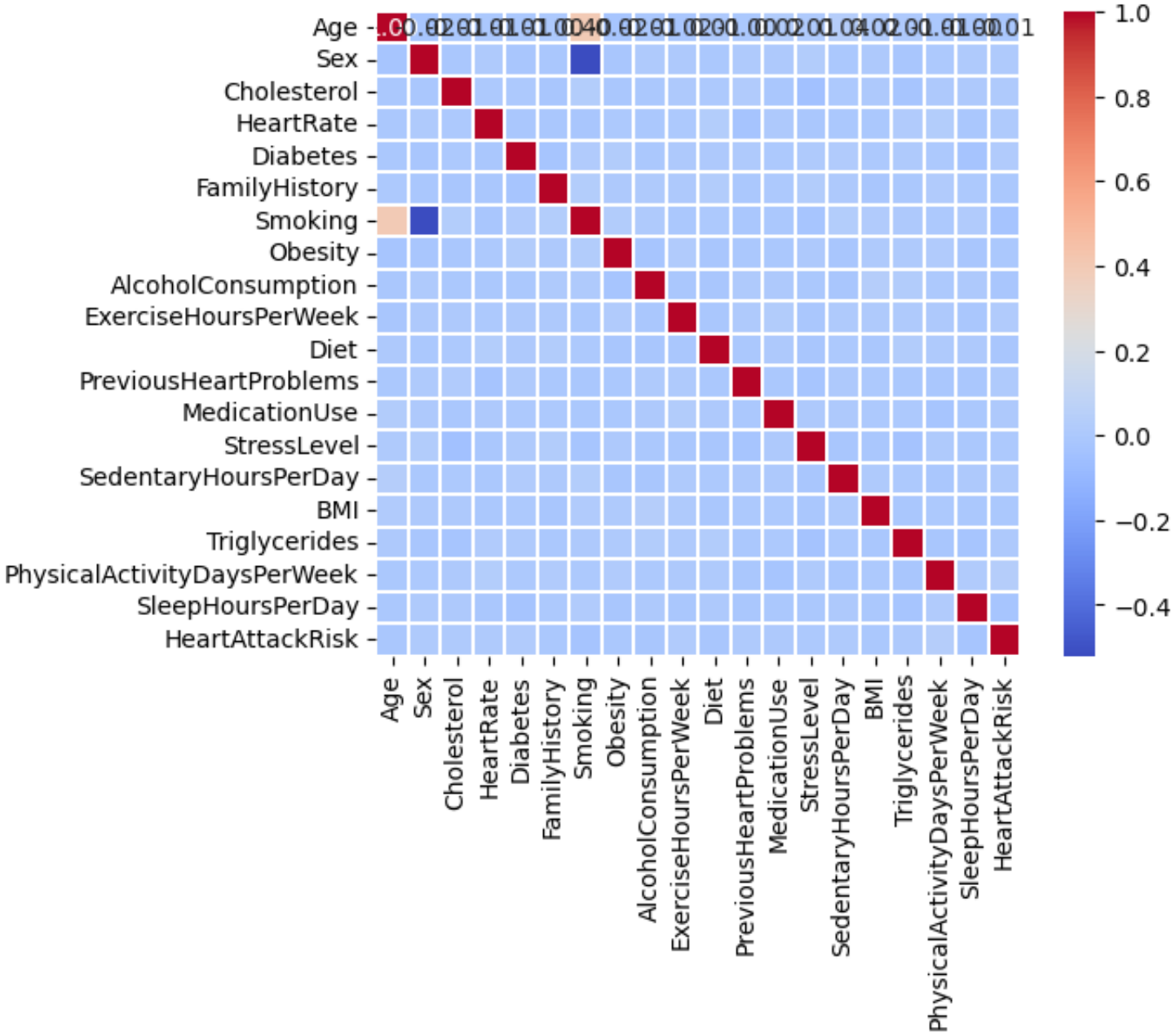
	StressLevel	SedentaryHoursPerDay	BMI
Age	0.009038	0.035277	0.015263
Sex	0.022613	-0.003197	0.007339
Cholesterol	-0.046384	0.002666	0.006640
HeartRate	-0.012162	-0.005735	-0.008719
Diabetes	0.013681	0.016116	0.001189
FamilyHistory	0.020897	0.008166	-0.016199
Smoking	-0.022779	0.027058	0.016838
Obesity	0.006992	-0.016499	0.009216
AlcoholConsumption	-0.003342	-0.020887	0.034459
ExerciseHoursPerWeek	-0.012457	0.013513	0.014652
Diet	-0.017945	0.000562	-0.013721
PreviousHeartProblems	-0.022240	0.005068	-0.000679
MedicationUse	0.006701	0.008403	0.002361
StressLevel	1.000000	-0.020169	-0.011685
SedentaryHoursPerDay	-0.020169	1.000000	0.009772
BMI	-0.011685	0.009772	1.000000
Triglycerides	-0.034245	-0.002160	-0.009687
PhysicalActivityDaysPerWeek	0.009616	-0.010272	0.000037
SleepHoursPerDay	-0.004619	0.006039	-0.005013
HeartAttackRisk	0.001721	0.011001	-0.004451

	Triglycerides	PhysicalActivityDaysPerWeek
Age	0.005797	-0.008239
Sex	-0.019952	-0.002054
Cholesterol	-0.021990	0.004431
HeartRate	0.019355	0.027651
Diabetes	0.014343	0.000773
FamilyHistory	-0.004002	0.017197
Smoking	0.007147	-0.002384
Obesity	0.011983	0.019373

AlcoholConsumption	0.022087	0.001501
ExerciseHoursPerWeek	0.013066	0.005985
Diet	0.022980	0.014075
PreviousHeartProblems	-0.004783	-0.009203
MedicationUse	-0.000727	-0.022539
StressLevel	-0.034245	0.009616
SedentaryHoursPerDay	-0.002160	-0.010272
BMI	-0.009687	0.000037
Triglycerides	1.000000	-0.018427
PhysicalActivityDaysPerWeek	-0.018427	1.000000
SleepHoursPerDay	-0.024415	0.020184
HeartAttackRisk	0.003151	0.033341

	SleepHoursPerDay	HeartAttackRisk
Age	-0.004341	-0.011901
Sex	0.013754	0.011592
Cholesterol	0.008683	0.012943
HeartRate	-0.014396	0.013301
Diabetes	-0.027086	0.018105
FamilyHistory	0.000234	-0.007273
Smoking	0.019011	-0.027275
Obesity	-0.019675	-0.006824
AlcoholConsumption	0.007448	-0.016818
ExerciseHoursPerWeek	-0.012526	0.019948
Diet	0.007391	-0.019790
PreviousHeartProblems	-0.010219	0.010103
MedicationUse	-0.004811	0.005181
StressLevel	-0.004619	0.001721
SedentaryHoursPerDay	0.006039	0.011001
BMI	-0.005013	-0.004451
Triglycerides	-0.024415	0.003151
PhysicalActivityDaysPerWeek	0.020184	0.033341
SleepHoursPerDay	1.000000	-0.022781
HeartAttackRisk	-0.022781	1.000000

Out[22]: <Axes: >


Age
Sex
Cholesterol
HeartRate
Diabetes
FamilyHistory
Smoking
Obesity
AlcoholConsumption
ExerciseHoursPerWeek
Diet
PreviousHeartProblems
MedicationUse
StressLevel
SedentaryHoursPerDay
BMI
Triglycerides
PhysicalActivityDaysPerWeek
SleepHoursPerDay
HeartAttackRisk
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

In [23]:

```
# # plotar o heatmap para as variáveis numéricas
# # Corrigido
# correlacaoOriginal = trainOriginal.corr(numeric_only=True)
# print('\nCORRELAÇÃO: \n', correlacaoOriginal)
# sns.heatmap(correlacaoOriginal, cmap='coolwarm', fmt='.2f', linewidths=0.1, vmax=1.0, square=True, linecolor='wh
```

In [24]:

```
# train_idx = train.shape[0]
# print('TRAIN: ')
# print(train.head())
# test_idx = test.shape[0]

# print('\nTESTE: ')
# print(test.head())

# target = train.HeartAttackRisk.copy()
# train.drop(['HeartAttackRisk'], axis=1, inplace=True)

# # concatenar treino e teste em um único DataFrame
# df_merged = pd.concat(objs=[train, test], axis=0).reset_index(drop=True)
# print('\n\ndf_merged.shape: ({} x {})'.format(df_merged.shape[0], df_merged.shape[1]))
```

In [25]:

```
# df_merged.head()
# print(df_merged.head())
# df_mergedOriginal = df_merged
```

In [26]:

```
#df_merged.isnull().sum()
```

In [27]:
```
# # age
# age_median = df_merged['Age'].median()
# df_merged['Age'].fillna(age_median, inplace=True)

# # obesity
# obesity_median = df_merged['Obesity'].median()
# df_merged['Obesity'].fillna(obesity_median, inplace=True)

# # identificar a maior ocorrencia
# risk_top = df_merged['HeartAttackRisk'].value_counts().index[0]
# # substituir ausencias pela maior ocorrencia
# df_merged['HeartAttackRisk'].fillna(risk_top, inplace=True)
```

In [28]:
```
# # recuperar datasets de traino e teste
# # train_idx contem o numero de amostras do dataset: train
# print('NUMERO DE AMOSTRAS NO DATASET ORIGINAL: train  ',train_idx)
# train = df_merged.iloc[:train_idx]

# display(train.head())
# display(test.head)
# display(target.head())
```

1. Codificação One-Hot:

Converte variáveis categóricas (por exemplo, sexo, tabagismo) em representações numéricas usando codificação one-hot para que possam ser usadas em modelos de aprendizado de máquina.

In [29]:
```
# Codificação One-Hot para variáveis categóricas
encoder = OneHotEncoder(handle_unknown='ignore')
features_to_encode = ['Sex', 'Smoking', 'AlcoholConsumption', 'Diet', 'PreviousHeartProblems', 'MedicationUse']
encoder.fit(train[features_to_encode])
```

Out[29]:
```
▾ OneHotEncoder
OneHotEncoder(handle_unknown='ignore')
```

In [30]:
```
# Transformar os dados de treinamento e teste
train_encoded = encoder.transform(train[features_to_encode]).toarray()
test_encoded = encoder.transform(test[features_to_encode]).toarray()
```

In [31]:
```
# Renomear as colunas codificadas
new_column_names = encoder.get_feature_names_out(features_to_encode)
train_encoded = pd.DataFrame(train_encoded, columns=new_column_names)
test_encoded = pd.DataFrame(test_encoded, columns=new_column_names)
```

In [32]:
```
# Cocatenar od dados codifcados aos DataFrames originais
train = pd.concat([train, train_encoded], axis=1)
test = pd.concat([test, test_encoded], axis=1)
```

In [33]:
```
# Removendo as colunas originais após a codificação
train.drop(features_to_encode, axis=1, inplace=True)
test.drop(features_to_encode, axis=1, inplace=True)
```

In [34]:
```
# Separando o alvo dos dados de treinamento
target = train['HeartAttackRisk']
train.drop(['HeartAttackRisk'], axis=1, inplace=True)
```

1. Imputação de Valores Ausentes:

Trata valores ausentes nos dados substituindo-os pela mediana dos valores da coluna.

In [35]:
```
# Imputação de valores ausentes com a mediana
for column in train.columns:
    train[column].fillna(train[column].median(), inplace=True)
    test[column].fillna(train[column].median(), inplace=True)
```

1. Divisão e Treinamento do Modelo:

- Divide os dados de treinamento em conjuntos de treinamento e validação para avaliar o desempenho do modelo.
- Cria e treina um modelo de Árvore de Decisão para prever o risco de ataque cardíaco.

In [36]:
```
# Dividindo os dados de treinamento para validação cruzada
X_train, X_val, y_train, y_val = train_test_split(train, target, test_size=0.2, random_state=42)
```

In [37]:
```
# Converter nomes de colunas para string
X_train.columns = X_train.columns.astype(str)
X_val.columns = X_val.columns.astype(str)
```

In [38]:
```
# Criando e treinando o modelo de árvore de decisão
tree_model = DecisionTreeClassifier(max_depth=9, criterion='entropy')
tree_model.fit(X_train, y_train)
```

Out[38]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=9)
```

In [39]:
```
# Avaliando o modelo no conjunto de validação
y_pred_tree_val = tree_model.predict(X_val)
accuracy_val = accuracy_score(y_val, y_pred_tree_val)
print('Acurácia no conjunto de validação:', accuracy_val)
```

```
Acurácia no conjunto de validação: 0.61
```

In [40]:
```
# Remover a coluna 'HeartAttackRisk' do DataFrame test
test.drop(['HeartAttackRisk'], axis=1, inplace=True)
```

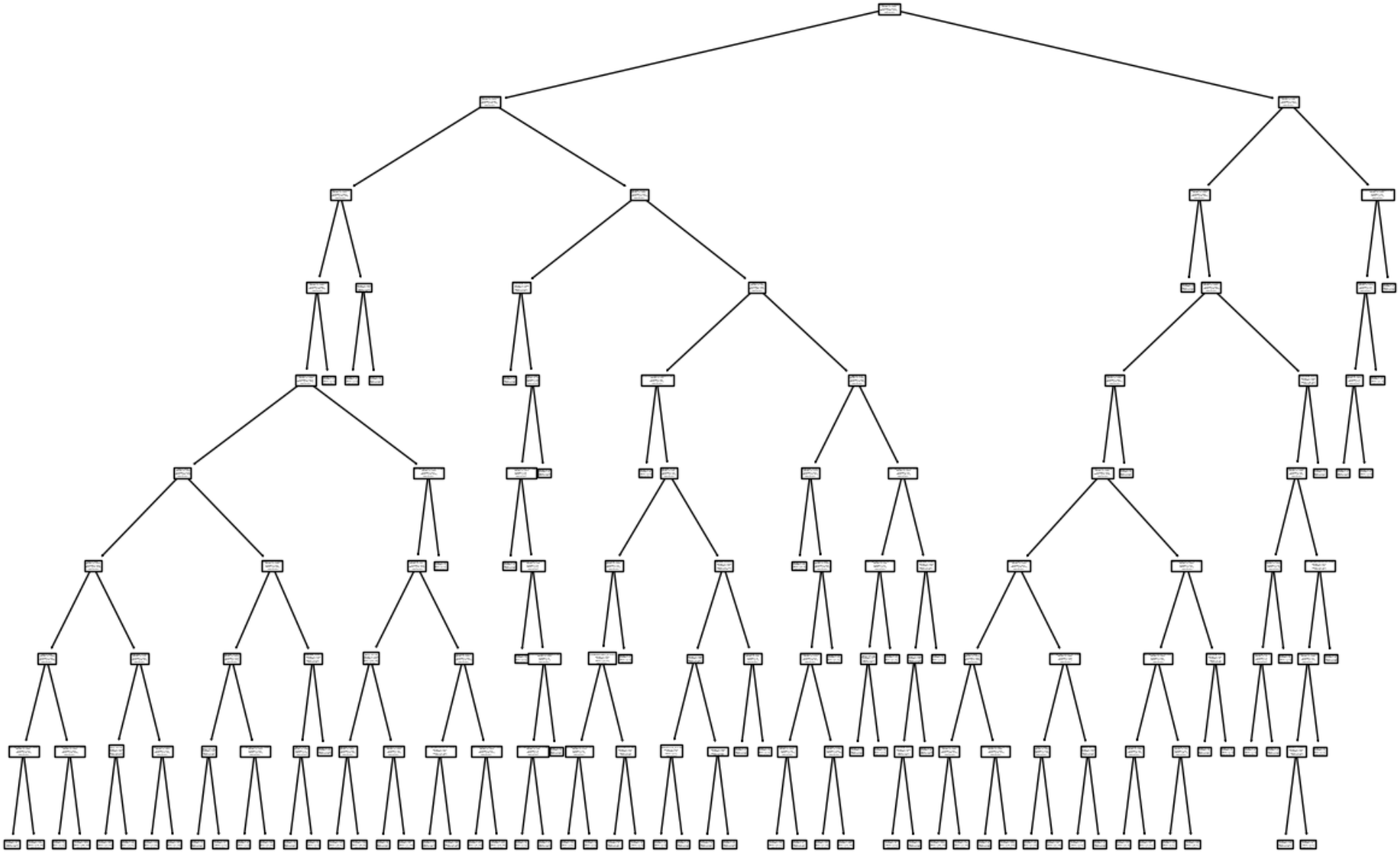
1. Avaliação e Previsões do Modelo:

- Avalia o desempenho do modelo nos conjuntos de validação e teste usando a métrica de acurácia.
- Faz previsões no conjunto de teste usando o modelo treinado.

In [41]:
```
# Fazendo previsões no conjunto de teste
y_pred_tree = tree_model.predict(test)
accuracy_test = accuracy_score(np.array(y_test['HeartAttackRisk']), y_pred_tree)
print('Acurácia no conjunto de teste:', accuracy_test)
```

```
Acurácia no conjunto de teste: 0.6230820547031354
```

In [42]:
```
# Plotando a árvore de decisão
fig = plt.figure(figsize=(15, 10))
_ = tree.plot_tree(tree_model, feature_names=train.columns, class_names={0: 'Não', 1: 'Sim'})
```



1. Função de Previsão:

Define uma função para receber informações do paciente, realizar as transformações necessárias nos dados e fornecer uma previsão de risco de ataque cardíaco usando o modelo treinado.

In [43]:

```
def prever_risco():
    sexo = int(input("Digite o sexo (0: Feminino, 1: Masculino): "))
    idade = int(input("Digite a idade: "))
    fuma = int(input("Fuma? (0: Não, 1: Sim: )"))
    obesidade = int(input("Obeso? (0: Não, 1: Sim): "))
    alcool = int(input("Consome Alcool? (0: Não, 1: Não): "))
    diabetes = int(input("Diabético? (0: Não, 1: Sim): "))
    dieta = int(input("Dieta (0: Saudável, 1: Não Saudável, 2: Moderada): "))
    problemas_cardiacos = int(input("Problemas cardíacos prévios? (0: Não, 1: Sim): "))
    uso_medicacao = int(input("Uso de medicação? (0: Não, 1: Sim): "))
    colesterol = int(input("Taxa de Colesterol: " ))
    # Criando um DataFrame com os dados do paciente
    dados_entrada = {'Age': [idade],
                     'Cholesterol': [colesterol],#[train['Cholesterol'].median()], # Valor da mediana do treino
                     'HeartRate': [train['HeartRate'].median()],    # Valor da mediana do treino
                     'Diabetes': [diabetes],
                     'FamilyHistory': [train['FamilyHistory'].median()],  # Valor da mediana do treino
                     'Obesity': [obesidade],
                     'ExerciseHoursPerWeek': [train['ExerciseHoursPerWeek'].median()], # Valor da mediana do trein
                     'StressLevel': [train['StressLevel'].median()],   # Valor da mediana do treino
                     'SedentaryHoursPerDay': [train['SedentaryHoursPerDay'].median()], # Valor da mediana do trein
                     'BMI': [train['BMI'].median()],     # Valor da mediana do treino
                     'Triglycerides': [train['Triglycerides'].median()],  # Valor da mediana do treino
                     'PhysicalActivityDaysPerWeek': [train['PhysicalActivityDaysPerWeek'].median()], # Valor da me
                     'SleepHoursPerDay': [train['SleepHoursPerDay'].median()], # Valor da mediana do treino
                     'Sex': [sexo], 'Smoking': [fuma],
                     'AlcoholConsumption': [alcool],
                     'Diet': [dieta], 'PreviousHeartProblems': [problemas_cardiacos],
                     'MedicationUse': [uso_medicacao]}

    pessoa = pd.DataFrame(dados_entrada)

    # Codificando as variáveis categóricas
    pessoa_encoded = encoder.transform(pessoa[features_to_encode]).toarray()
```

```
    new_column_names = encoder.get_feature_names_out(features_to_encode)
    pessoa_encoded = pd.DataFrame(pessoa_encoded, columns=new_column_names)

    # Concatenar os dados codificados ao DataFrame original
    pessoa = pd.concat([pessoa, pessoa_encoded], axis=1)

    # Remover as colunas originais após a codificação
    pessoa.drop(features_to_encode, axis=1, inplace=True)

    # Converter nomes de colunas para string
    pessoa.columns = pessoa.columns.astype(str)

    # Fazendo a previsão
    resultado = tree_model.predict(pessoa)

    if resultado == 0:
        print("\nPaciente sem risco de infarto.")
    else:
        print("\nPaciente com risco de infarto.")
prever_risco()
```

```
Paciente com risco de infarto.
```