Introduction to Data Science (S2-22_DSECLZG532)-ASSIGNMENT

Group No 141

Group Member Names:

1. PORURI V S HARI HARA NANDAN
2. SIBEN NAYAK

1. Business Understanding

Students are expected to identify an analytical problem of your choice. You have to detail the Business Understanding part of your problem under this heading which basically addresses the following questions.

1. What is the business problem that you are trying to solve?
2. What data do you need to answer the above problem?
3. What are the different sources of data?
4. What kind of analytics task are you performing?

Score: 1 Mark in total (0.25 mark each)

----------Type the answers below this line----------

1. Cardiovascular diseases have been rising since the past few years. We see news of people dying almost every other day due to heart attacks. This project involves analyzing a healthcare dataset with the aim of predicting the prognosis of heart disease, so that we can predict and possib;y prevent heart attacks in time and save lives.

2. The dataset includes various features related to patients' health and lifestyle, including age, sex, general health, checkup frequency, exercise habits, smoking history, and the presence of various diseases. Each entry represents a unique patient, and the features capture various factors associated with disease prognosis.

3. We have used data from public domain. The Behavioral Risk Factor Surveillance System (BRFSS) is the nation’s premier system of health-related telephone surveys that collect state data about U.S. residents regarding their health-related risk behaviors, chronic health conditions, and use of preventive services. From 304 unique variables in the BRFSS dataset, we picked 19 variables that relate to lifestyle factors of a person that can be contributed to being at risk with any form of Cardiovascular Diseases.

4. Intially we will be performing explorartory data analysis, feature engineering, observe different paramters of columns using bivarite analysis which affects the health of individuals. Later on, we will use feature engineering and apply some ML models to understand the top features and train our systems with the data available for these features so that it can accurately predict hear disease.

Examples: We'll explore the relationship between each variable and the target variable (Heart_Disease). This analysis will allow us to understand how each variable is associated with the presence or absence of heart disease. We can use techniques like bar charts to visualize the distributions of the target variable based on different categories or levels of other variables.

2. Data Acquisition

For the problem identified , find an appropriate data set (Your data set must be unique with minimum **20 features and 10k rows**) from any public data source.

2.1 Download the data directly

In [ ]:
```
## importing necessary libraries ###

import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np

# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns

import urllib.request
```

In [ ]:
```
##---------Type the code below this line-----------------##
request_url = urllib.request.urlopen('https://raw.githubusercontent.com/Nandan8593/datascience/master/CVD_raw.csv')
```

2.2 Code for converting the above downloaded data into a dataframe

In [ ]:
```
##---------Type the code below this line-----------------##
#dataset=pd.read_csv("CVD_cleaned.csv")
dataset=pd.read_csv(request_url)
```

2.3 Confirm the data has been correctly by displaying the first 5 and last 5 records.

In [ ]:
```
##---------Type the code below this line-----------------##
### Display 1st 5 records ######
dataset.head(5)
```

Out[ ]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabetes	Arthritis	Sex	Age_Category	Height_(cm)	Weight_(kg)	BMI	Smoking_History	Alcohol_Consumption	Fruit_Consumption
0	Poor	Within the past 2 years	No	No	No	No	No	No	Yes	Female	70-74	150	32.66	14.54	Yes	0	30
1	Very Good	Within the past year	No	Yes	No	No	No	Yes	No	Female	70-74	165	77.11	28.29	No	0	30
2	Very Good	Within the past year	Yes	No	No	No	No	Yes	No	Female	60-64	163	88.45	33.47	No	4	12
3	Poor	Within the past year	Yes	Yes	No	No	No	Yes	No	Male	75-79	180	93.44	28.73	No	0	30
4	Good	Within the past year	No	No	No	No	No	No	No	Male	80+	191	88.45	24.37	Yes	0	8

In [ ]:
```
#### Display last 5 records #####
dataset.tail(5)
```

Out[ ]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabetes	Arthritis	Sex	Age_Category	Height_(cm)	Weight_(kg)	BMI	Smoking_History	Alcohol_Consumption	Fruit_Consu
308850	Very Good	Within the past year	Yes	No	No	No	No	No	No	Male	25-29	168	81.65	29.05	No	4	
308851	Fair	Within the past 5 years	Yes	No	No	No	No	Yes	No	Male	65-69	180	69.85	21.48	No	8	
308852	Very Good	5 or more years ago	Yes	No	No	No	Yes	Yes, but female told only during pregnancy	No	Female	30-34	157	61.23	24.69	Yes	4	
308853	Very Good	Within the past year	Yes	No	No	No	No	No	No	Male	65-69	183	79.38	23.73	No	3	
308854	Excellent	Within the past year	Yes	No	No	No	No	No	No	Female	45-49	160	81.19	31.71	No	1	

2.4 Display the column headings, statistical information, description and statistical summary of the data.

In [ ]:
```
##---------Type the code below this line-----------------##
## list of column headings ##
for col in dataset.columns:
    print(col)
```

```
General_Health
Checkup
Exercise
Heart_Disease
Skin_Cancer
Other_Cancer
Depression
Diabetes
Arthritis
Sex
Age_Category
Height_(cm)
Weight_(kg)
BMI
Smoking_History
Alcohol_Consumption
Fruit_Consumption
Green_Vegetables_Consumption
FriedPotato_Consumption
```

In [ ]:
```
## Information about the dataset
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 308855 entries, 0 to 308854
Data columns (total 19 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   General_Health                308855 non-null  object 
 1   Checkup                       308855 non-null  object 
 2   Exercise                      308855 non-null  object 
 3   Heart_Disease                 308855 non-null  object 
 4   Skin_Cancer                   308855 non-null  object 
 5   Other_Cancer                  308855 non-null  object 
 6   Depression                    308855 non-null  object 
 7   Diabetes                      308855 non-null  object 
 8   Arthritis                     308855 non-null  object 
 9   Sex                           308855 non-null  object 
 10  Age_Category                  308855 non-null  object 
 11  Height_(cm)                   308855 non-null  int64  
 12  Weight_(kg)                   308836 non-null  float64
 13  BMI                           308846 non-null  float64
 14  Smoking_History               308855 non-null  object 
 15  Alcohol_Consumption           308855 non-null  int64  
 16  Fruit_Consumption             308855 non-null  int64  
 17  Green_Vegetables_Consumption  308855 non-null  int64  
 18  FriedPotato_Consumption       308855 non-null  int64  
dtypes: float64(2), int64(5), object(12)
memory usage: 44.8+ MB
```

In [ ]:
```
## Showing the statistical information of the numerical variables
numeric_data = dataset.select_dtypes(include = ['number']) # Selects numeric datatypes only (We are doing this for statistical summary)
numeric_data.describe()
```

Out[ ]:

	Height_(cm)	Weight_(kg)	BMI	Alcohol_Consumption	Fruit_Consumption	Green_Vegetables_Consumption	FriedPotato_Consumption
count	308855.000000	308836.000000	308846.000000	308855.000000	308855.000000	308855.000000	308855.000000
mean	170.615230	83.588636	28.626216	5.096359	29.835201	15.110418	6.296621
std	10.658014	21.343307	6.522312	8.199750	24.875695	14.926220	8.582941
min	91.000000	24.950000	12.020000	0.000000	0.000000	0.000000	0.000000
25%	163.000000	68.040000	24.210000	0.000000	12.000000	4.000000	2.000000
50%	170.000000	81.650000	27.440000	1.000000	30.000000	12.000000	4.000000
75%	178.000000	95.250000	31.850000	6.000000	30.000000	20.000000	8.000000
max	241.000000	293.020000	99.330000	30.000000	120.000000	128.000000	128.000000

In [ ]:
```
## Showing the statistical information of the categorical variables
categorical_data = dataset.select_dtypes(exclude = ['number']) # Selects categorical datatypes only (We are doing this for statistical summary)
categorical_data.describe()
```

Out[ ]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabetes	Arthritis	Sex	Age_Category	Smoking_History
count	308855	308855	308855	308855	308855	308855	308855	308855	308855	308855	308855	308855
unique	5	5	2	2	2	2	2	4	2	2	13	2
top	Very Good	Within the past year	Yes	No	No	No	No	No	No	Female	65-69	No
freq	110395	239372	239382	283884	278861	278977	246954	259142	207783	160196	33434	183590

Observations (Statistical Summary)

1. **Health Perception and Physical Attributes**:
    - A considerable number of individuals perceive their general health as "Very Good".
    - The average height and weight across the dataset suggest a population with diverse physical characteristics. The average BMI, however, falls into the "overweight" category, which could be a potential area of concern.
2. **Proactivity Towards Health**:
    - The majority have had a medical checkup within the past year, signaling a population that's proactive about health monitoring.
3. **Active Lifestyle but with Dietary Concerns**:
    - Most respondents engage in exercise, suggesting an active lifestyle.
    - Despite the active lifestyle, some dietary habits, such as fried potato consumption, might be areas to probe further for potential health implications.
4. **Low Disease Prevalence but High-Risk Indicators**:
    - Most do not have heart disease, skin cancer, other forms of cancer, or diabetes. However, the average BMI is in the overweight range, which is a known risk factor for many conditions.
    - While many don't smoke, there's still a significant portion of smokers, which is a crucial risk factor for cardiovascular diseases.
5. **Age and Gender Dynamics**:
    - A notable proportion of the dataset belongs to the elderly group, especially in the "65-69" age bracket. Elderly populations have unique health risks and needs.
    - There's a slight dominance of females in the dataset. Given that certain health conditions and responses can vary by gender, this skew might be significant for deeper analyses.
6. **Potential Areas of Concern**:
    - The prevalence of conditions like depression and arthritis signals underlying health challenges that might not be immediately life-threatening but can significantly affect the quality of life.
    - The average alcohol consumption is noteworthy, with some individuals consuming alcohol up to 30 times, which could have implications for liver health and other conditions.
7. **Disease Distribution and Other Health Conditions**:
    - The majority of individuals in the dataset are free from the conditions listed. This is optimistic but also poses challenges for predictive modeling due to potential class imbalances.
8. **Rich Dataset with Diverse Indicators**:
    - The dataset encompasses a range of health indicators, from physical attributes like height, weight, and BMI to behaviors like exercise and smoking, to conditions like heart disease and depression. This variety allows for a holistic view of health and can be pivotal for in-depth analyses.

2.5 Write your observations from the above.

1. Size of the dataset
2. What type of data attributes are there?
3. Is there any null data that has to be cleaned?

Score: 2 Marks in total (0.25 marks for 2.1, 0.25 marks for 2.2, 0.5 marks for 2.3, 0.25 marks for 2.4, 0.75 marks for 2.5)

In [ ]:
```
dataset.shape
```

Out[ ]:
```
(308855, 19)
```

In [ ]:
```
print("The dataset has the following data attributes")
dataset.info()
```

```
The dataset has the following data attributes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 308855 entries, 0 to 308854
Data columns (total 19 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   General_Health                308855 non-null  object 
 1   Checkup                       308855 non-null  object 
 2   Exercise                      308855 non-null  object 
 3   Heart_Disease                 308855 non-null  object 
 4   Skin_Cancer                   308855 non-null  object 
 5   Other_Cancer                  308855 non-null  object 
 6   Depression                    308855 non-null  object 
 7   Diabetes                      308855 non-null  object 
 8   Arthritis                     308855 non-null  object 
 9   Sex                           308855 non-null  object 
 10  Age_Category                  308855 non-null  object 
 11  Height_(cm)                   308855 non-null  int64  
 12  Weight_(kg)                   308836 non-null  float64
 13  BMI                           308846 non-null  float64
 14  Smoking_History               308855 non-null  object 
 15  Alcohol_Consumption           308855 non-null  int64  
 16  Fruit_Consumption             308855 non-null  int64  
 17  Green_Vegetables_Consumption  308855 non-null  int64  
 18  FriedPotato_Consumption       308855 non-null  int64  
dtypes: float64(2), int64(5), object(12)
memory usage: 44.8+ MB
```

In [ ]:
```
print("Checking for null values")
dataset.isnull().sum()
```

```
Checking for null values
```

Out[ ]:
```
General_Health                  0
Checkup                         0
Exercise                        0
Heart_Disease                   0
Skin_Cancer                     0
Other_Cancer                    0
Depression                      0
Diabetes                        0
Arthritis                       0
Sex                             0
Age_Category                    0
Height_(cm)                     0
Weight_(kg)                    19
BMI                             9
Smoking_History                 0
Alcohol_Consumption             0
Fruit_Consumption               0
Green_Vegetables_Consumption    0
FriedPotato_Consumption         0
dtype: int64
```

----------Type the answers below this line----------

Observations

1. The dataset has 308855 rows and 19 columns.
2. The dataset has categorical and numerical data attributes.
3. The dataset has 19 rows of Weight_(kg) as null and 9 rows of BMI as null.

3. Data Preparation

If input data is numerical or categorical, do 3.1, 3.2 and 3.4 If input data is text, do 3.3 and 3.4

3.1 Check for

- duplicate data
- missing data
- data inconsistencies

In [ ]:
```
##---------Type the code below this line-----------------##
print("Total number of duplicates:", dataset.duplicated().sum())
```

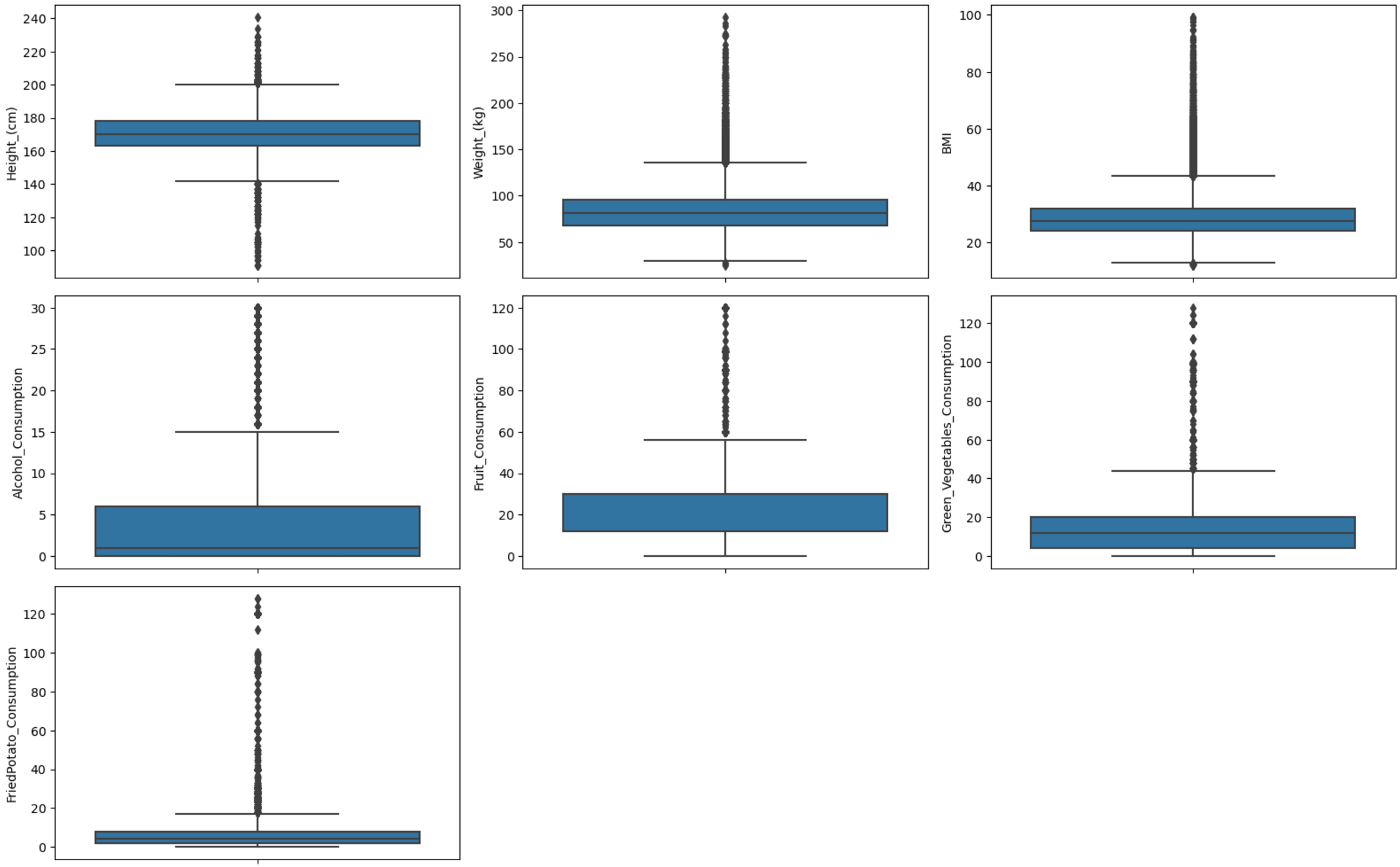
```
Total number of duplicates: 80
```

In [ ]:
```
print("Checking for missing data:", dataset.isnull().sum())
```

```
Checking for missing data: General_Health                  0
Checkup                           0
Exercise                          0
Heart_Disease                     0
Skin_Cancer                       0
Other_Cancer                      0
Depression                        0
Diabetes                          0
Arthritis                         0
Sex                               0
Age_Category                      0
Height_(cm)                       0
Weight_(kg)                      19
BMI                               9
Smoking_History                   0
Alcohol_Consumption               0
Fruit_Consumption                 0
Green_Vegetables_Consumption      0
FriedPotato_Consumption           0
dtype: int64
```

In [ ]:
```
print("Checking for Outliers")
# Create box plots
plt.figure(figsize=(16, 10))
for i, column in enumerate(numeric_data, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(y=dataset[column])

plt.tight_layout()
```

```
Checking for Outliers
```



Interpretation of Results:

The summary statistics and boxplots indicate that there are some potential outliers in our numerical data. Here are a few observations:

- Height_(cm): The minimum value is 91 cm, and the maximum is 241 cm. These could be extreme cases, but they're worth investigating further.
- Weight_(kg): The maximum weight is 293.02 kg, which seems quite high. This could potentially be an outlier or extreme value.

- BMI: The maximum BMI is 99.33, which is very high, even for extreme cases of obesity. This might indicate data entry errors.
- Alcohol_Consumption: The maximum value is 30, which seems quite high. We need to understand the measurement units to interpret whether this is an outlier or not.
- Fruit_Consumption, Green_Vegetables_Consumption, FriedPotato_Consumption: The maximum values seem quite high, but it depends on the measurement units (for example, servings per week/month).

These potential outliers and extreme values should be further investigated to determine their validity and possible impact on the analysis.

3.2 Apply techniques

- to remove duplicate data
- to impute or remove missing data
- to remove data inconsistencies

In [ ]:
```
##---------Type the code below this line-----------------##
print("Dropping Duplicate Data")
dataset.drop_duplicates()
```

```
Dropping Duplicate Data
```

Out[ ]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabetes	Arthritis	Sex	Age_Category	Height_(cm)	Weight_(kg)	BMI	Smoking_History	Alcohol_Consumption	Fruit_Consu
0	Poor	Within the past 2 years	No	No	No	No	No	No	Yes	Female	70-74	150	32.66	14.54	Yes	0	
1	Very Good	Within the past year	No	Yes	No	No	No	Yes	No	Female	70-74	165	77.11	28.29	No	0	
2	Very Good	Within the past year	Yes	No	No	No	No	Yes	No	Female	60-64	163	88.45	33.47	No	4	
3	Poor	Within the past year	Yes	Yes	No	No	No	Yes	No	Male	75-79	180	93.44	28.73	No	0	
4	Good	Within the past year	No	No	No	No	No	No	No	Male	80+	191	88.45	24.37	Yes	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
308850	Very Good	Within the past year	Yes	No	No	No	No	No	No	Male	25-29	168	81.65	29.05	No	4	
308851	Fair	Within the past 5 years	Yes	No	No	No	No	Yes	No	Male	65-69	180	69.85	21.48	No	8	
308852	Very Good	5 or more years ago	Yes	No	No	No	Yes	Yes, but told only during pregnancy	No	Female	30-34	157	61.23	24.69	Yes	4	
308853	Very Good	Within the past year	Yes	No	No	No	No	No	No	Male	65-69	183	79.38	23.73	No	3	
308854	Excellent	Within the past year	Yes	No	No	No	No	No	No	Female	45-49	160	81.19	31.71	No	1	

308775 rows × 19 columns

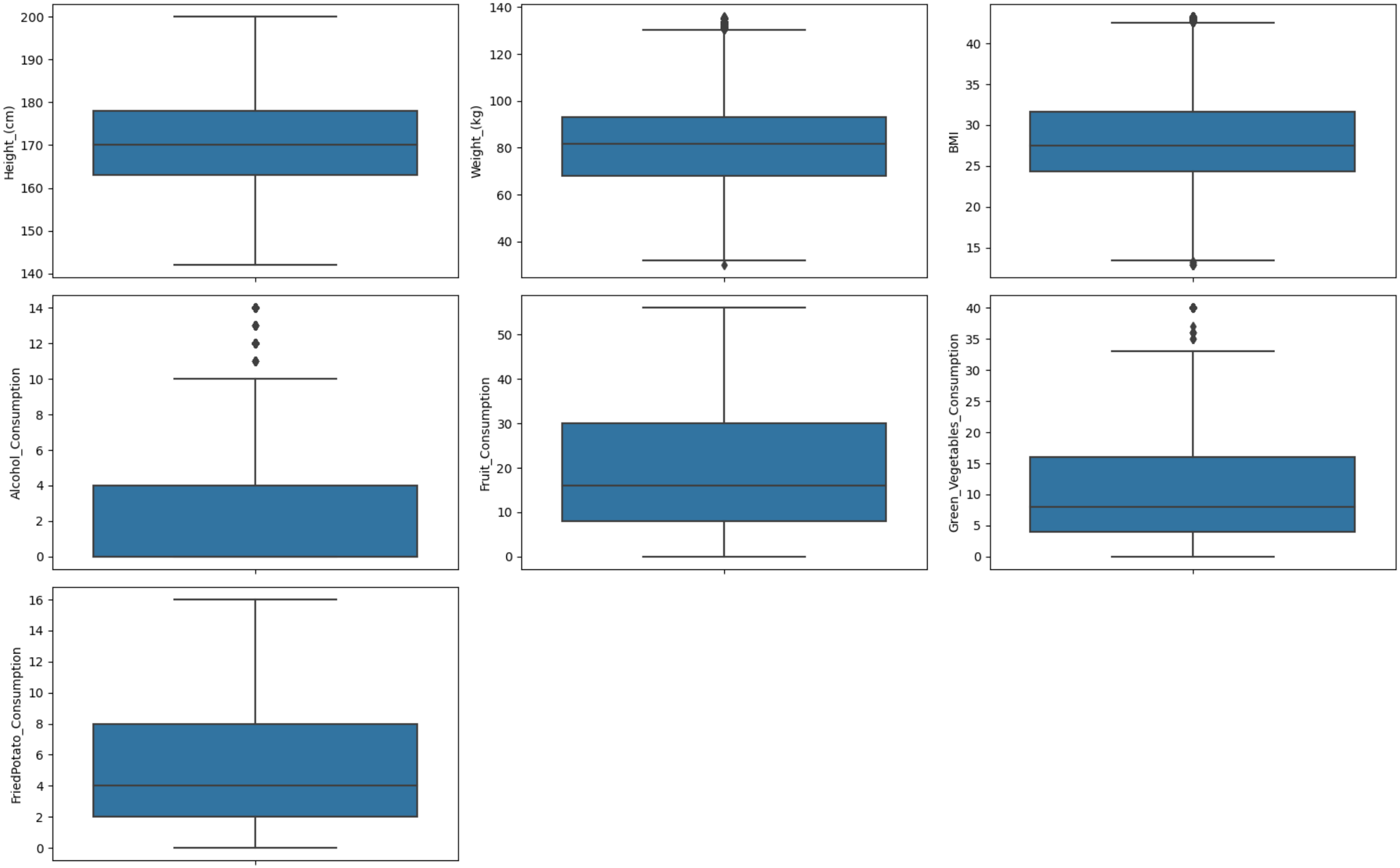
In [ ]:
```
print("Imputing Missing Data")
dataset['BMI']=dataset['BMI'].fillna(dataset["Weight_(kg)"]/dataset["Height_(cm)"]/dataset["Height_(cm)"]*10000)
dataset['Weight_(kg)']=dataset['Weight_(kg)'].fillna(dataset['BMI']**dataset["Height_(cm)"]**dataset["Height_(cm)"]/10000)
dataset.isnull().sum()
```

```
Imputing Missing Data
```

Out[ ]:
```
General_Health                0
Checkup                       0
Exercise                      0
Heart_Disease                 0
Skin_Cancer                   0
Other_Cancer                  0
Depression                    0
Diabetes                      0
Arthritis                     0
Sex                           0
Age_Category                  0
Height_(cm)                   0
Weight_(kg)                   0
BMI                           0
Smoking_History               0
Alcohol_Consumption           0
Fruit_Consumption             0
Green_Vegetables_Consumption  0
FriedPotato_Consumption       0
dtype: int64
```

In [ ]:
```
print("Removing Data Inconsistencies")
outlier_indices = set()  # Set to store outlier indices

for col in numeric_data:
    Q1 = dataset[col].quantile(0.25)
    Q3 = dataset[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5*IQR
    upper = Q3 + 1.5*IQR

    # Add outlier indices to the set
    upper_array = np.where(dataset[col] >= upper)[0]
    lower_array = np.where(dataset[col] <= lower)[0]
    outlier_indices.update(upper_array)
    outlier_indices.update(lower_array)

# Drop outlier rows
dataset.drop(index=outlier_indices, axis=0, inplace=True)

# Create box plots
plt.figure(figsize=(16, 10))
for i, column in enumerate(numeric_data, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(y=dataset[column])

plt.tight_layout()
```

```
Removing Data Inconsistencies
```



3.3 Encode categorical data

In [ ]:
```
# Mapping for Diabetes
diabetes_mapping = {
    'No': 0,
    'No, pre-diabetes or borderline diabetes': 0,
    'Yes, but female told only during pregnancy': 1,
    'Yes': 1
```

```
}
dataset['Diabetes'] = dataset['Diabetes'].map(diabetes_mapping)

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

# Convert remaining categorical variables with "Yes" and "No" values to binary format for correlation computation
binary_columns = ['Heart_Disease', 'Skin_Cancer', 'Other_Cancer', 'Depression', 'Arthritis', 'Smoking_History','Exercise']

for column in binary_columns:
    dataset[column] = dataset[column].map({'Yes': 1, 'No': 0})

# Ordinal encoding for General_Health, Checkup, Age_Category
general_health_mapping = {
    'Poor': 0,
    'Fair': 1,
    'Good': 2,
    'Very Good': 3,
    'Excellent': 4
}
dataset['General_Health'] = dataset['General_Health'].map(general_health_mapping)

checkup_mapping = {
    'Never': 0,
    '5 or more years ago': 0.2,
    'Within the past 5 years': 1,
    'Within the past 2 years': 2,
    'Within the past year': 4

}
dataset['Checkup'] = dataset['Checkup'].map(checkup_mapping)

age_category_mapping = {
    '18-24': 0,
    '25-29': 1,
    '30-34': 2,
    '35-39': 3,
    '40-44': 4,
    '45-49': 5,
    '50-54': 6,
    '55-59': 7,
    '60-64': 8,
    '65-69': 9,
    '70-74': 10,
    '75-79': 11,
    '80+': 12
}
dataset['Age_Category'] = dataset['Age_Category'].map(age_category_mapping)

dataset
```

Out[ ]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabetes	Arthritis	Sex	Age_Category	Height_(cm)	Weight_(kg)	BMI	Smoking_History	Alcohol_Consumption	Fruit_Consumpti
0	0	2.0	0	0	0	0	0	0	1	1	10	150	32.66	14.54	1	0	
1	3	4.0	0	1	0	0	0	1	0	1	10	165	77.11	28.29	0	0	
2	3	4.0	1	0	0	0	0	1	0	1	8	163	88.45	33.47	0	4	
3	0	4.0	1	1	0	0	0	1	0	0	11	180	93.44	28.73	0	0	
4	2	4.0	0	0	0	0	0	0	0	0	12	191	88.45	24.37	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
308849	2	1.0	1	0	0	0	0	0	0	0	7	168	58.97	20.98	0	0	
308850	3	4.0	1	0	0	0	0	0	0	0	1	168	81.65	29.05	0	4	
308852	3	0.2	1	0	0	0	1	1	0	1	2	157	61.23	24.69	1	4	
308853	3	4.0	1	0	0	0	0	0	0	0	9	183	79.38	23.73	0	3	
308854	4	4.0	1	0	0	0	0	0	0	1	5	160	81.19	31.71	0	1	

180107 rows × 19 columns

3.4 Text data

1. Remove special characters
2. Change the case (up-casing and down-casing).
3. Tokenization — process of discretizing words within a document.
4. Filter Stop Words.

In [ ]:
```
##---------Type the code below this line-----------------##
```

3.4 Report

Mention and justify the method adopted

- to remove duplicate data, if present
- to impute or remove missing data, if present
- to remove data inconsistencies, if present

OR for textdata

- How many tokens after step 3?
- how may tokens after stop words filtering?

If the any of the above are not present, then also add in the report below.

Score: 2 Marks (based on the dataset you have, the data prepreation you had to do and report typed, marks will be distributed between 3.1, 3.2, 3.3 and 3.4)

---------Type the code below this line-----------------

1. Our data is basically numerical and categorical data, we have observed the duplicates using `duplicated()` method. Later,in the next step we have dropped duplicates using `drop_duplicates()`
2. We have missing data for the fields BMI and Weights*(kg) columns, we have used the formula below to impute missing values: BMI = Weight*(kg)/ Height*(cm)/ Height*(cm) * 10000
3. We have computed quantiles for numerical attributes and from there we have plotted BOX PLOTS to understand and remove the outliers.

3.5 Identify the target variables.

- Separate the data from the target such that the dataset is in the form of (X,y) or (Features, Label)
- Discretize / Encode the target variable or perform one-hot encoding on the target or any other as and if required.
- Report the observations

Score: 1 Mark

In [ ]:
```
##---------Type the code below this line-----------------##
# Separate the data and target
X = dataset.drop('Heart_Disease', axis=1)  # This drops the 'Heart_Disease' column and keeps the rest as features
y = dataset['Heart_Disease']  # This selects the 'Heart_Disease' column as the target/label

X.head()
```

Out[ ]:

	General_Health	Checkup	Exercise	Skin_Cancer	Other_Cancer	Depression	Diabetes	Arthritis	Sex	Age_Category	Height_(cm)	Weight_(kg)	BMI	Smoking_History	Alcohol_Consumption	Fruit_Consumption	Green_Vegetables
0	0	2.0	0	0	0	0	0	1	1	10	150	32.66	14.54	1	0	30	
1	3	4.0	0	0	0	0	1	0	1	10	165	77.11	28.29	0	0	30	
2	3	4.0	1	0	0	0	1	0	1	8	163	88.45	33.47	0	4	12	
3	0	4.0	1	0	0	0	1	0	0	11	180	93.44	28.73	0	0	30	
4	2	4.0	0	0	0	0	0	0	0	12	191	88.45	24.37	1	0	8	

In [ ]:
```
df_target_encoded = pd.get_dummies(y, prefix='Heart_Disease')
df_target_encoded.head()
```

Out[ ]:

	Heart_Disease_0	Heart_Disease_1
0	True	False
1	False	True
2	True	False
3	False	True
4	True	False

Feature Engineering

In [ ]:
```
##---------Type the code below this line-----------------##
dataset_fe = dataset.copy()

# BMI Category
dataset_fe['BMI_Category'] = pd.cut(dataset_fe['BMI'], bins=[0, 18.5, 24.9, 29.9, np.inf], labels=['Underweight', 'Normal weight', 'Overweight', 'Obesity'])

# Health Checkup Frequency
```

```
checkup_mapping = {'Within the past year': 4, 'Within the past 2 years': 2, 'Within the past 5 years': 1, '5 or more years ago': 0.2, 'Never': 0}
dataset_fe['Checkup_Frequency'] = dataset_fe['Checkup'].replace(checkup_mapping)

# Lifestyle Score
exercise_mapping = {'Yes': 1, 'No': 0}
smoking_mapping = {'Yes': 1, 'No': 0}
dataset_fe['Lifestyle_Score'] = dataset_fe['Exercise'].replace(exercise_mapping) - dataset_fe['Smoking_History'].replace(smoking_mapping) + dataset_fe['Fruit_Consumption']/10 + dataset_fe['Green

# Healthy Diet Score
dataset_fe['Healthy_Diet_Score'] = dataset_fe['Fruit_Consumption']/10 + dataset_fe['Green_Vegetables_Consumption']/10 - dataset_fe['FriedPotato_Consumption']/10

# Interaction Terms
dataset_fe['Smoking_Alcohol'] = dataset_fe['Smoking_History'].replace(smoking_mapping) * dataset_fe['Alcohol_Consumption']
dataset_fe['Checkup_Exercise'] = dataset_fe['Checkup_Frequency'] * dataset_fe['Exercise'].replace(exercise_mapping)

# Ratio of Height to Weight
dataset_fe['Height_to_Weight'] = dataset_fe['Height_(cm)'] / dataset_fe['Weight_(kg)']

# Fruit and Vegetables Consumption Interaction
dataset_fe['Fruit_Vegetables'] = dataset_fe['Fruit_Consumption'] * dataset_fe['Green_Vegetables_Consumption']

# Healthy_Diet_Lifestyle Interaction
dataset_fe['HealthyDiet_Lifestyle'] = dataset_fe['Healthy_Diet_Score'] * dataset_fe['Lifestyle_Score']

# Alcohol_FriedPotato Interaction
dataset_fe['Alcohol_FriedPotato'] = dataset_fe['Alcohol_Consumption'] * dataset_fe['FriedPotato_Consumption']
```

Observations

BMI_Category: This variable categorizes the BMI (Body Mass Index) of each individual into one of four categories: 'Underweight', 'Normal weight', 'Overweight', and 'Obesity'. This is based on the following ranges: underweight is a BMI less than 18.5, normal weight is a BMI from 18.5 to 24.9, overweight is a BMI from 25 to 29.9, and obesity is a BMI of 30 or higher.

Checkup_Frequency: This variable represents the frequency of health check-ups for each individual. It assigns numeric values to the different frequency categories, where a higher value represents more frequent check-ups.

Lifestyle_Score: This variable provides a composite score based on various lifestyle factors including exercise, smoking, fruit consumption, green vegetable consumption, and alcohol consumption. Each lifestyle factor is assigned a certain weight, with positive activities like exercise and healthy eating contributing positively to the score, and negative activities like smoking and alcohol consumption subtracting from the score.

Healthy_Diet_Score: This variable calculates a score based on the individual's diet. It considers the consumption of fruits, green vegetables, and fried potatoes. More consumption of fruits and green vegetables adds positively to the score, while consumption of fried potatoes subtracts from the score.

Smoking_Alcohol: This interaction term represents the combination of smoking and alcohol consumption. It multiplies the mapped values of smoking history and alcohol consumption.

Checkup_Exercise: This interaction term represents the combination of health check-up frequency and exercise habits. It multiplies the mapped values of health check-up frequency and exercise habits.

Height_to_Weight: This variable calculates the ratio of an individual's height to their weight.

Fruit_Vegetables: This interaction term represents the combined consumption of fruits and green vegetables. It multiplies the values of fruit consumption and green vegetable consumption.

Depression_Exercise: This interaction term represents the combination of depression status and exercise habits. It multiplies the mapped values of depression status and exercise habits.

HealthyDiet_Lifestyle: This interaction term represents the combination of the Healthy Diet Score and the Lifestyle Score. It multiplies the values of these two scores.

Alcohol_FriedPotato: This interaction term represents the combined consumption of alcohol and fried potatoes. It multiplies the values of alcohol consumption and fried potato consumption.

4. Data Exploration using various plots

Feature Engineering

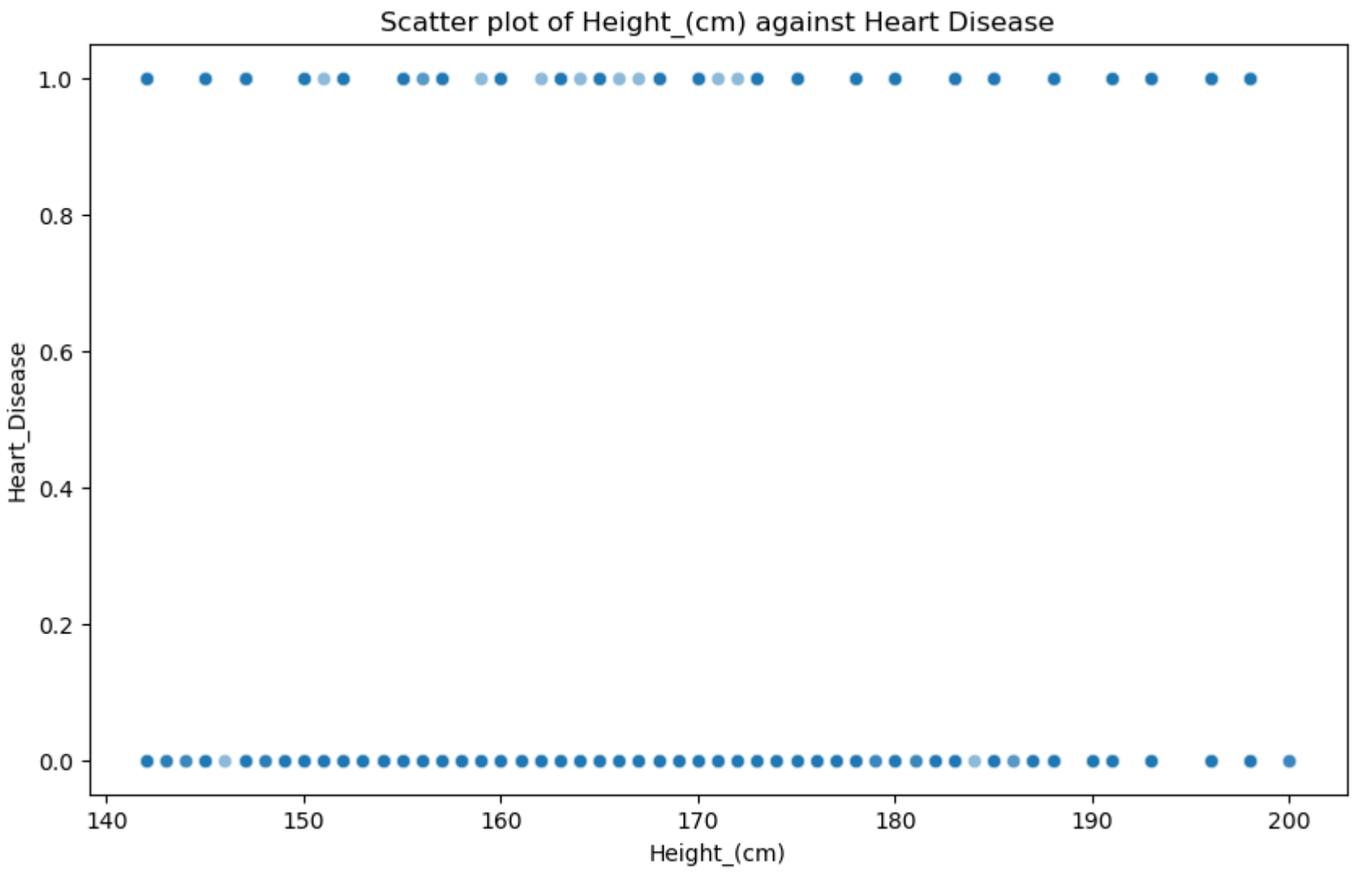
In [ ]:
```
##---------Type the code below this line-----------------##
dataset_sp = dataset.copy()

# BMI Category
dataset_sp['BMI_Category'] = pd.cut(dataset_sp['BMI'], bins=[0, 18.5, 24.9, 29.9, np.inf], labels=['Underweight', 'Normal weight', 'Overweight', 'Obesity'])

# Health Checkup Frequency
checkup_mapping = {'Within the past year': 4, 'Within the past 2 years': 2, 'Within the past 5 years': 1, '5 or more years ago': 0.2, 'Never': 0}
dataset_sp['Checkup_Frequency'] = dataset_sp['Checkup'].replace(checkup_mapping)

# Lifestyle Score
exercise_mapping = {'Yes': 1, 'No': 0}
smoking_mapping = {'Yes': 1, 'No': 0}
dataset_sp['Lifestyle_Score'] = dataset_sp['Exercise'].replace(exercise_mapping) - dataset_sp['Smoking_History'].replace(smoking_mapping) + dataset_sp['Fruit_Consumption']/10 + dataset_sp['Green

# Healthy Diet Score
dataset_sp['Healthy_Diet_Score'] = dataset_sp['Fruit_Consumption']/10 + dataset_sp['Green_Vegetables_Consumption']/10 - dataset_sp['FriedPotato_Consumption']/10

# Interaction Terms
dataset_sp['Smoking_Alcohol'] = dataset_sp['Smoking_History'].replace(smoking_mapping) * dataset_sp['Alcohol_Consumption']
dataset_sp['Checkup_Exercise'] = dataset_sp['Checkup_Frequency'] * dataset_sp['Exercise'].replace(exercise_mapping)

# Ratio of Height to Weight
dataset_sp['Height_to_Weight'] = dataset_sp['Height_(cm)'] / dataset_sp['Weight_(kg)']

# Fruit and Vegetables Consumption Interaction
dataset_sp['Fruit_Vegetables'] = dataset_sp['Fruit_Consumption'] * dataset_sp['Green_Vegetables_Consumption']

# Healthy_Diet_Lifestyle Interaction
dataset_sp['HealthyDiet_Lifestyle'] = dataset_sp['Healthy_Diet_Score'] * dataset_sp['Lifestyle_Score']

# Alcohol_FriedPotato Interaction
dataset_sp['Alcohol_FriedPotato'] = dataset_sp['Alcohol_Consumption'] * dataset_sp['FriedPotato_Consumption']
```

4.1 Scatter plot of each quantitative attribute with the target.

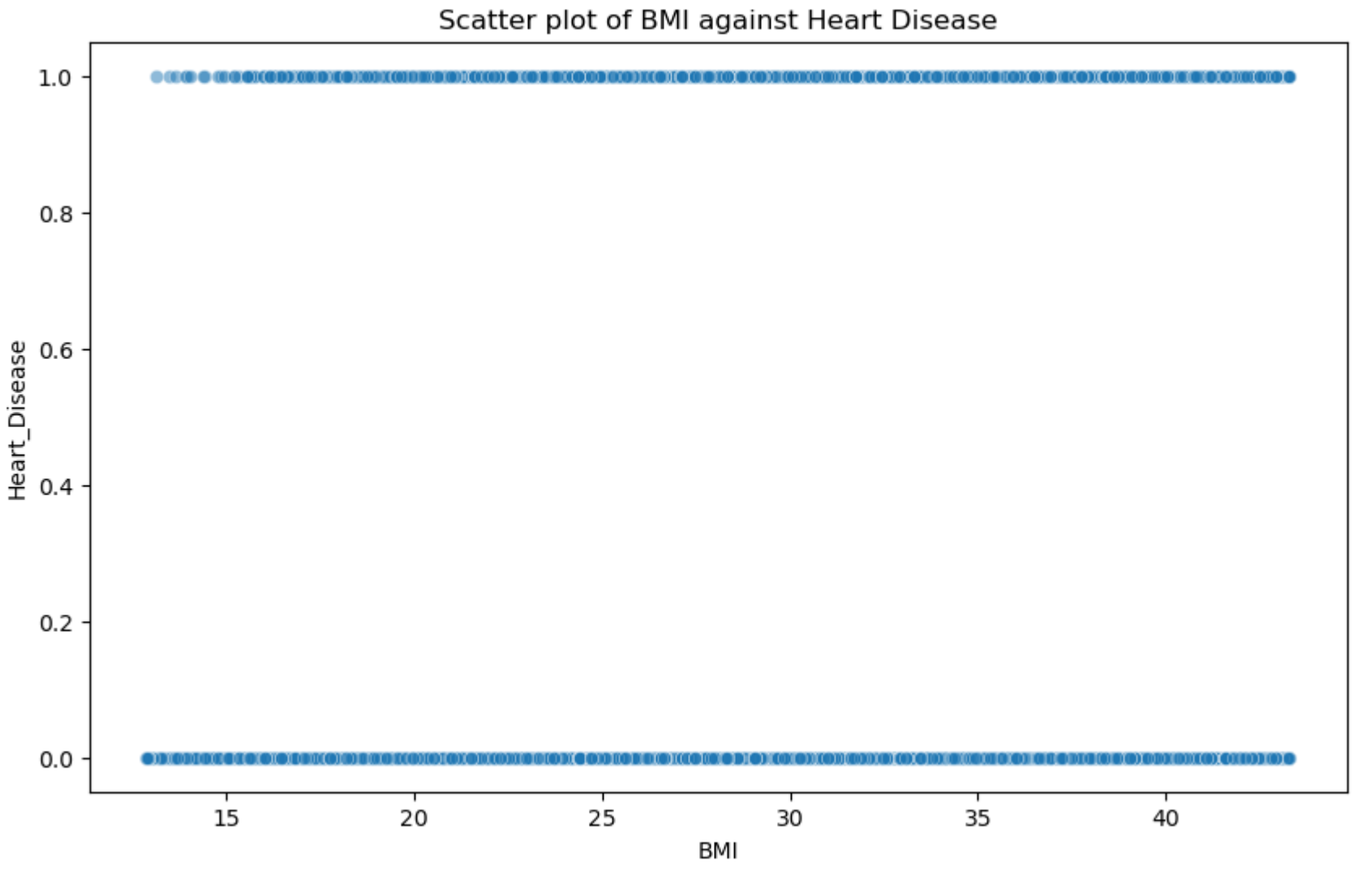
Score: 1 Mark

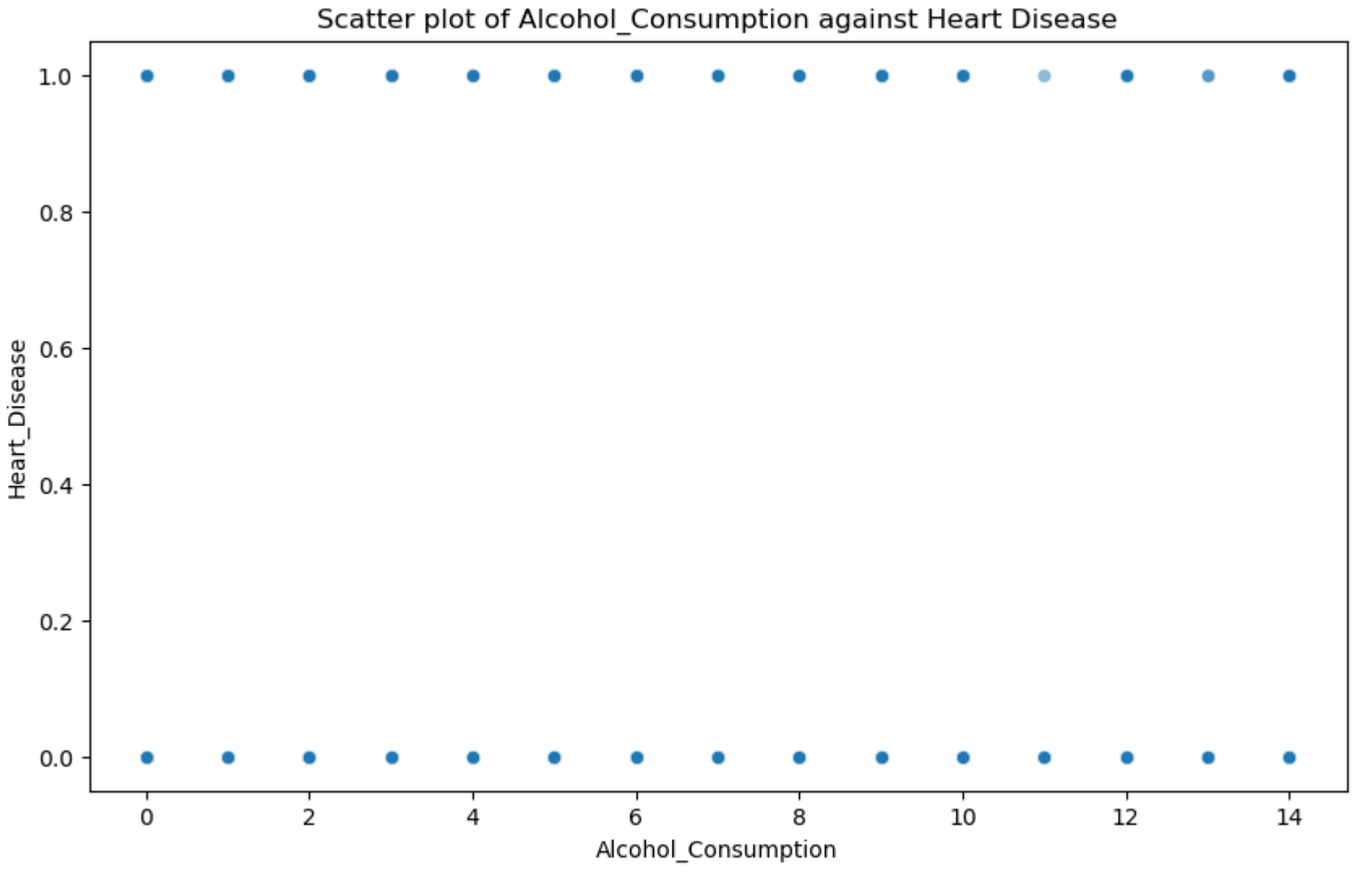
In [ ]:
```
# List of continuous features
continuous_features = ['Height_(cm)', 'Weight_(kg)', 'BMI', 'Alcohol_Consumption',
                       'Fruit_Consumption', 'Green_Vegetables_Consumption', 'FriedPotato_Consumption']

# List of categorical features
categorical_features = ['General_Health', 'Checkup', 'Exercise', 'Skin_Cancer',
                        'Other_Cancer', 'Depression', 'Diabetes', 'Arthritis', 'Sex', 'Age_Category', 'Smoking_History']

# Scatter plots for continuous features
for feature in continuous_features:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=dataset, x=feature, y='Heart_Disease', alpha=0.5)
    plt.title(f'Scatter plot of {feature} against Heart Disease')
    plt.show()

# Count plots for categorical features
for feature in categorical_features:
    plt.figure(figsize=(10, 6))
    sns.countplot(data=dataset, x=feature, hue='Heart_Disease')
```

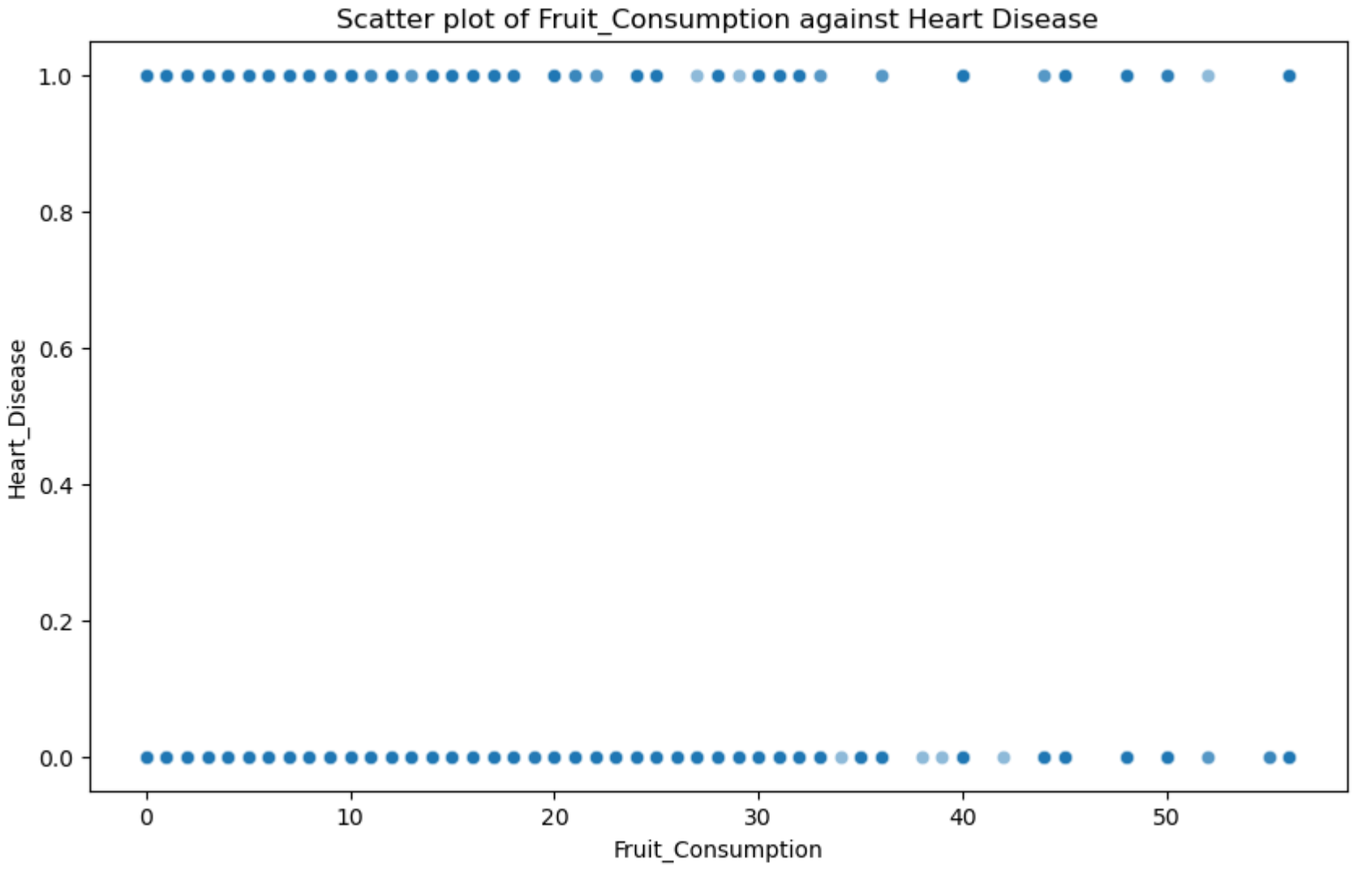
```
    plt.title(f'Count plot of {feature} against Heart Disease')
    plt.show()

combined_continuous_features = ['Lifestyle_Score', 'Healthy_Diet_Score', 'Height_to_Weight', 'Fruit_Vegetables', 'HealthyDiet_Lifestyle', 'Alcohol_FriedPotato']
combined_categorical_features = ['BMI_Category', 'Checkup_Frequency', 'Smoking_Alcohol', 'Checkup_Exercise']

# Scatter plots for combined features
for feature in combined_continuous_features:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=dataset_sp, x=feature, y='Heart_Disease', alpha=0.5)
    plt.title(f'Scatter plot of {feature} against Heart Disease')
    plt.show()

# Count plots for combined features
for feature in combined_categorical_features:
    plt.figure(figsize=(10, 6))
    sns.countplot(data=dataset_sp, x=feature, hue='Heart_Disease')
    plt.title(f'Count plot of {feature} against Heart Disease')
    plt.show()
```

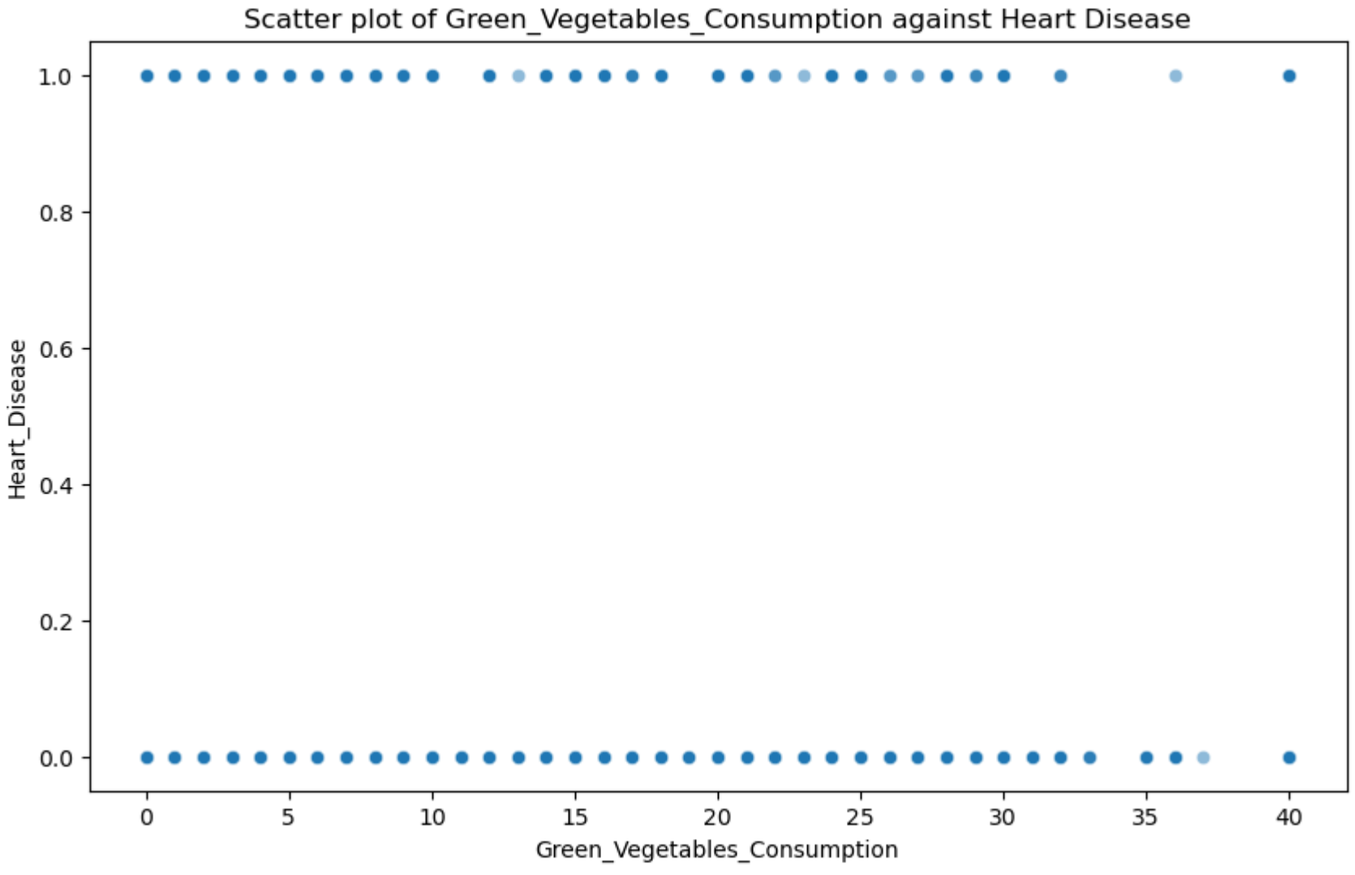


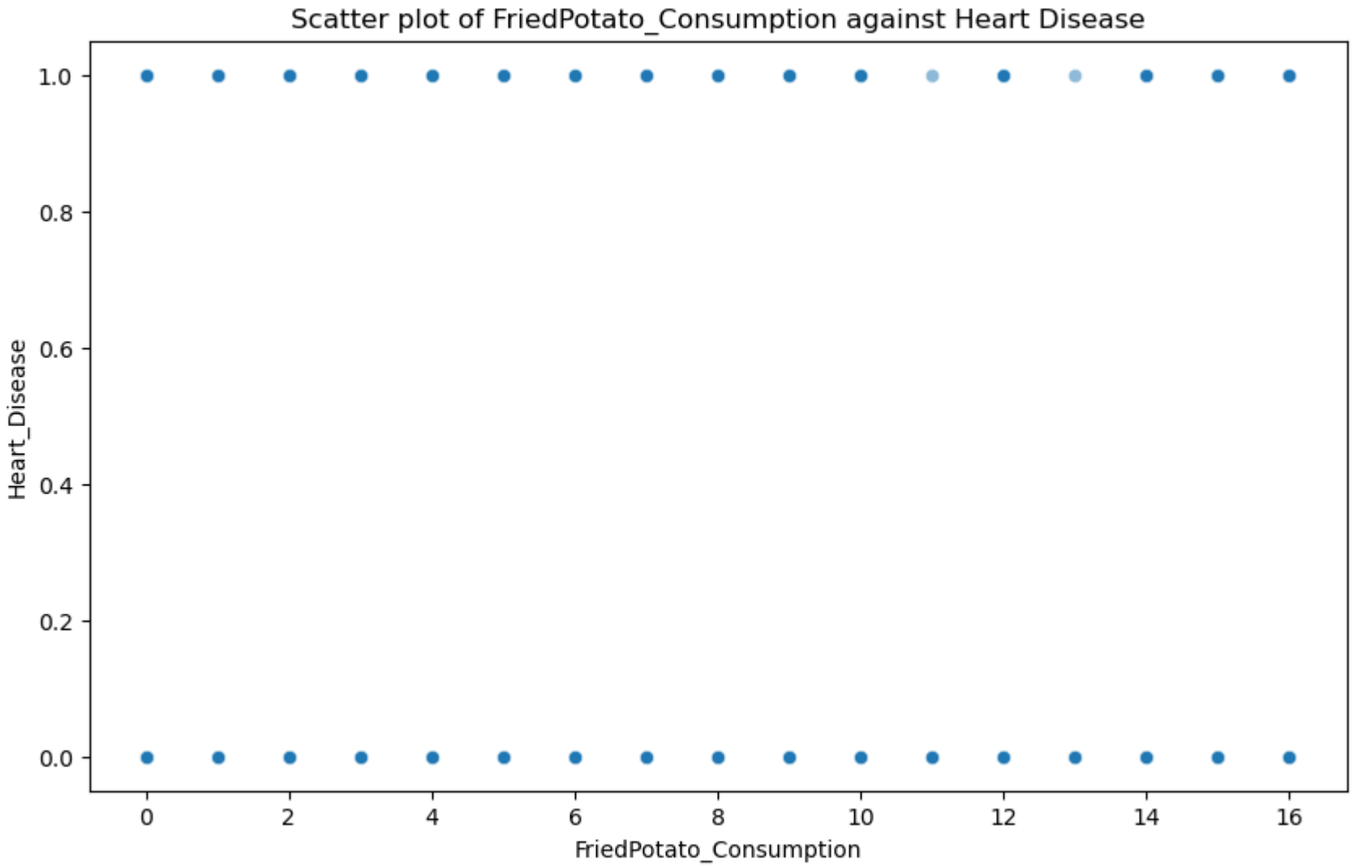
Scatter plot of Weight_(kg) against Heart Disease
Heart_Disease
Weight_(kg)
1.0
0.8
0.6
0.4
0.2
0.0
40
60
80
100
120
140


Scatter plot of BMI against Heart Disease
Heart_Disease
1.0
0.8
0.6
0.4
0.2
0.0
15
20
25
30
35
40
BMI

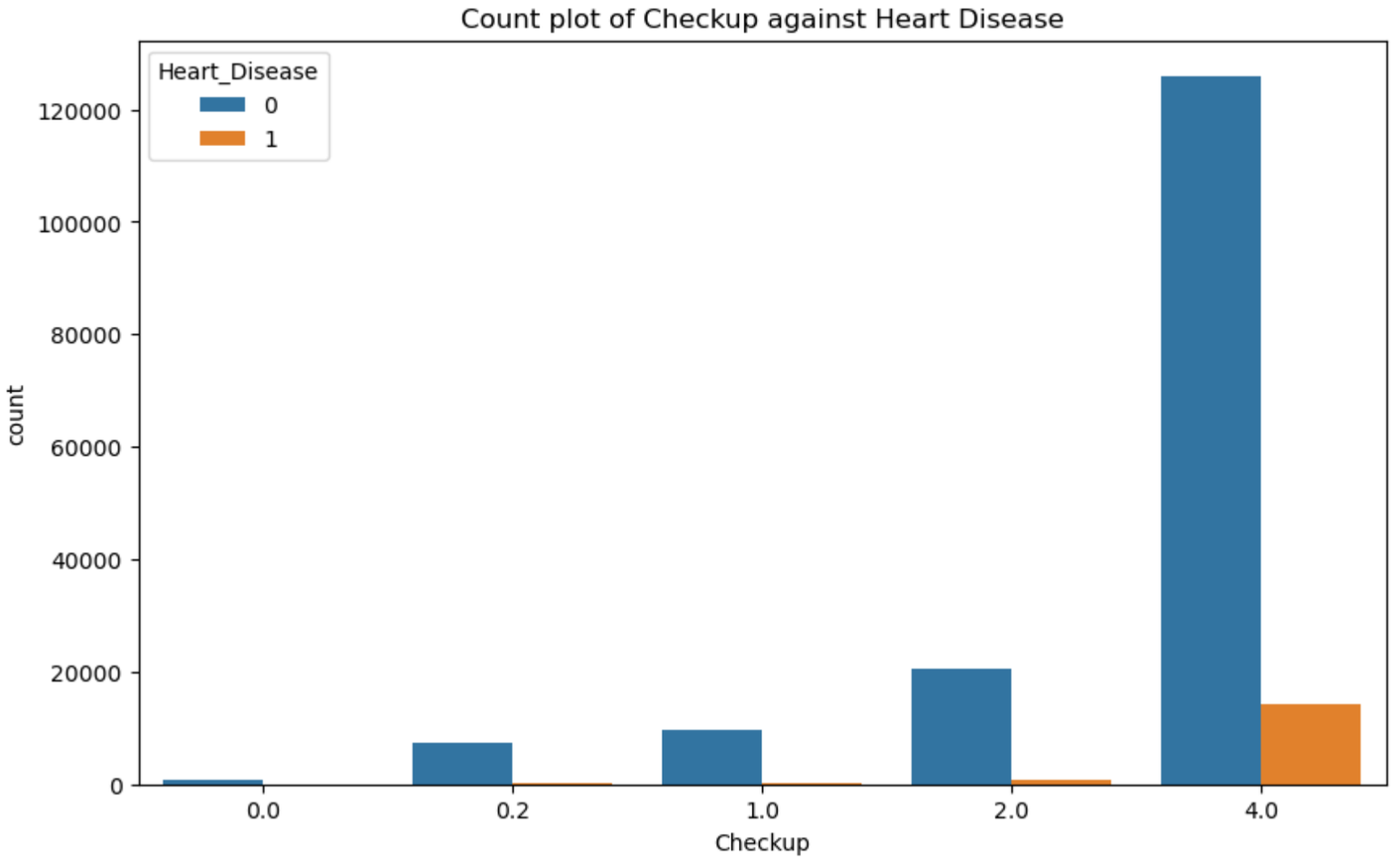

Scatter plot of Alcohol_Consumption against Heart Disease
Heart_Disease
Alcohol_Consumption
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14

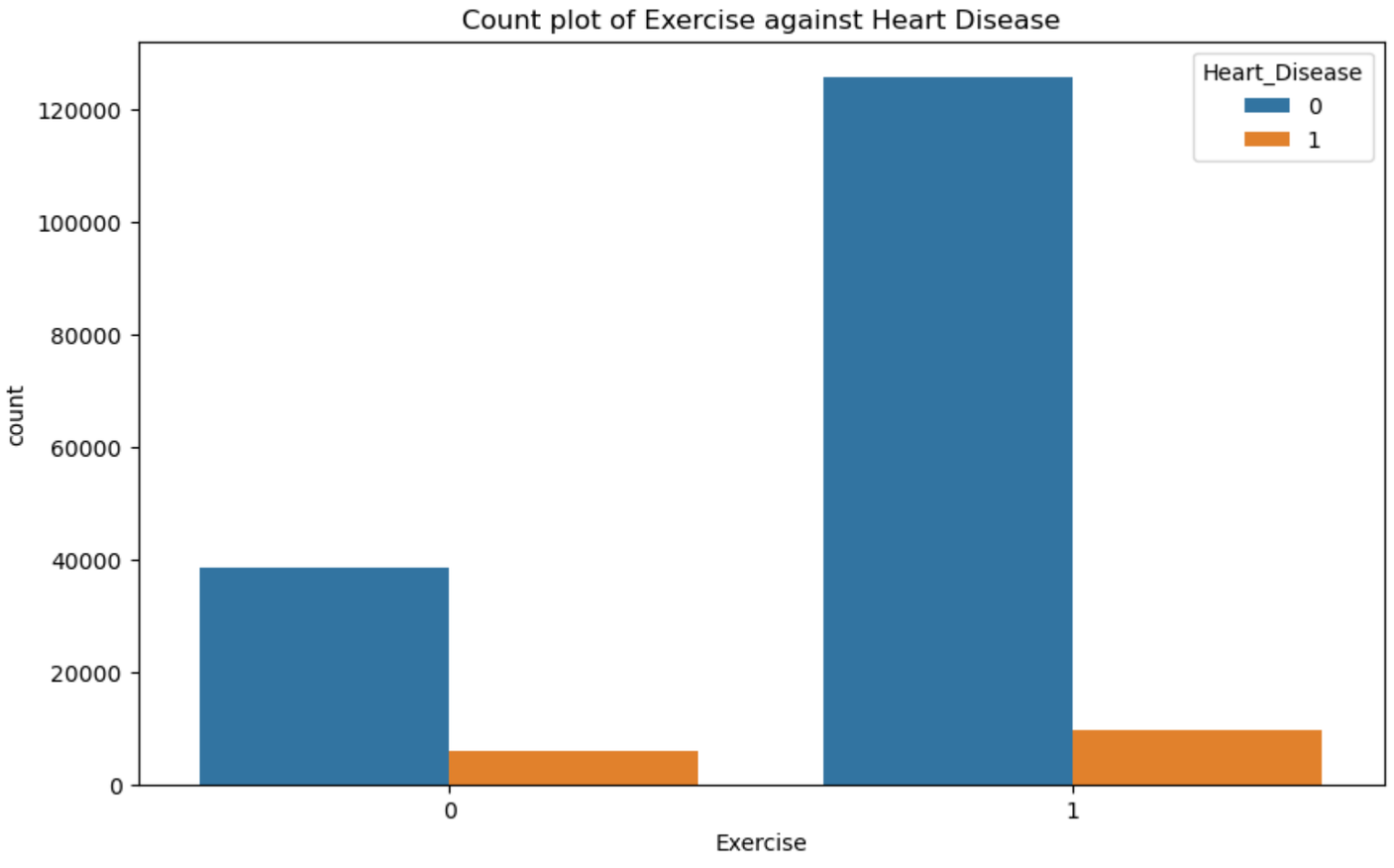

Scatter plot of Fruit_Consumption against Heart Disease
Heart_Disease
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
Fruit_Consumption

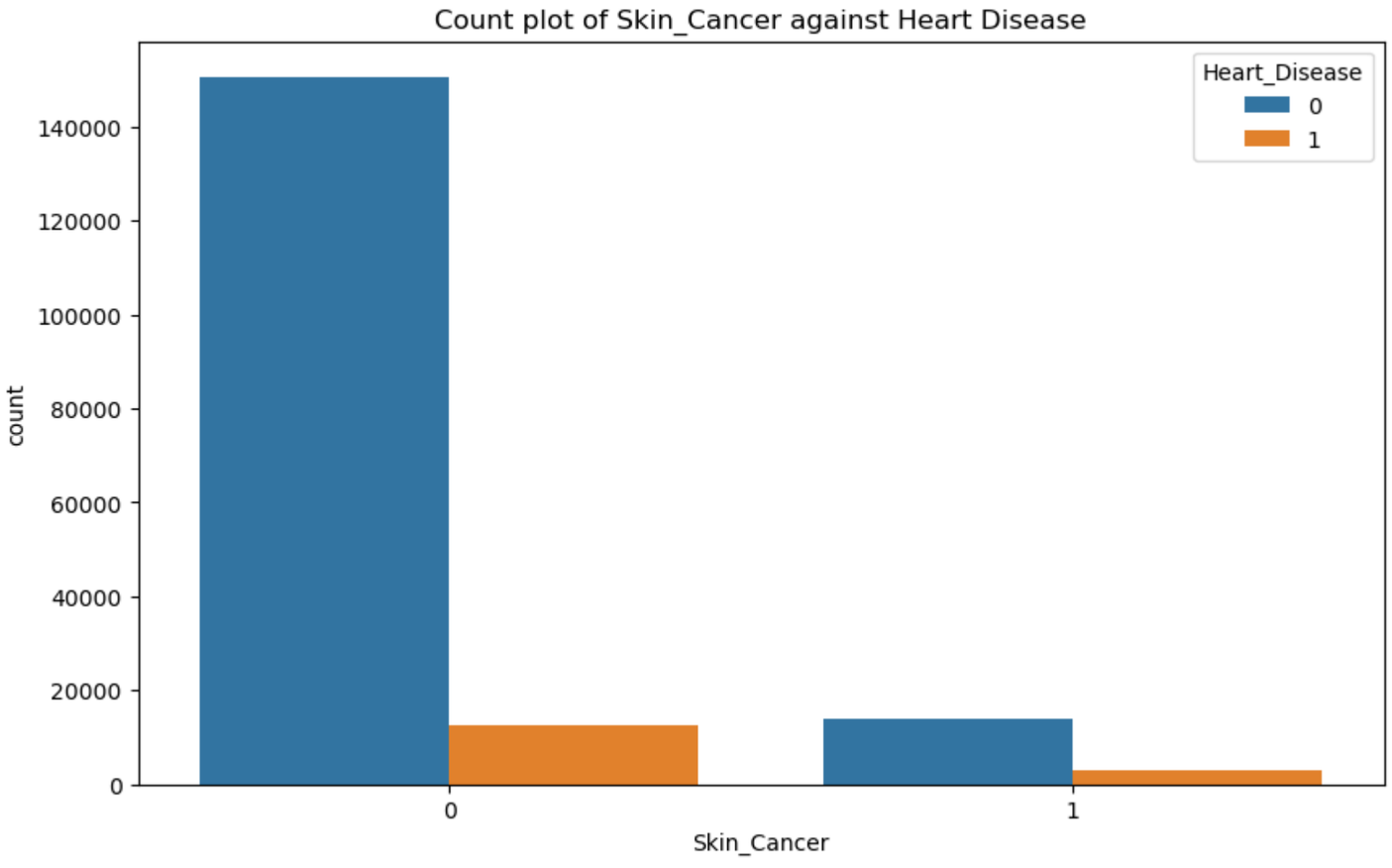

Scatter plot of Green_Vegetables_Consumption against Heart Disease
Heart_Disease
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
40
Green_Vegetables_Consumption

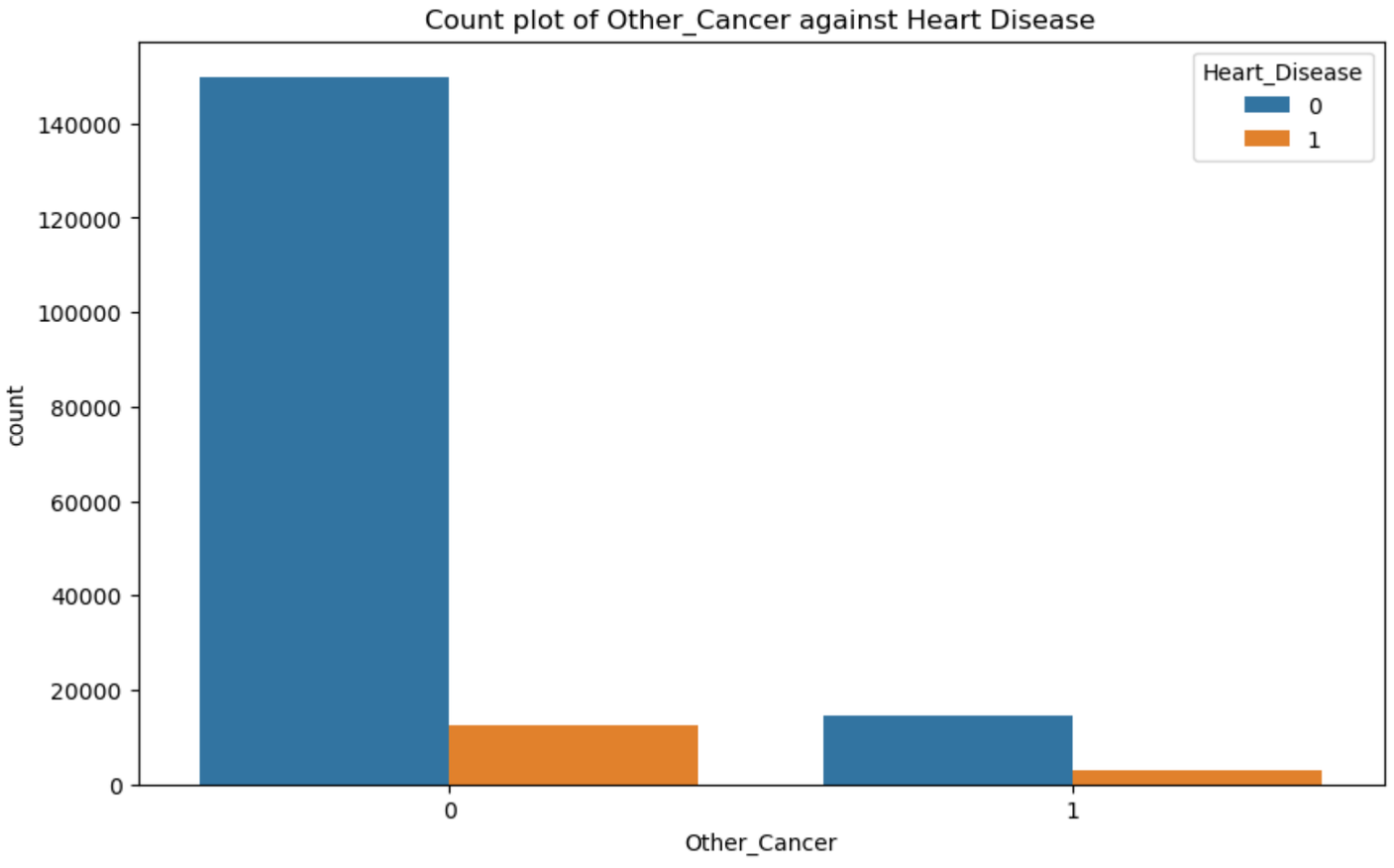

Scatter plot of FriedPotato_Consumption against Heart Disease
Heart_Disease
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14
16
FriedPotato_Consumption

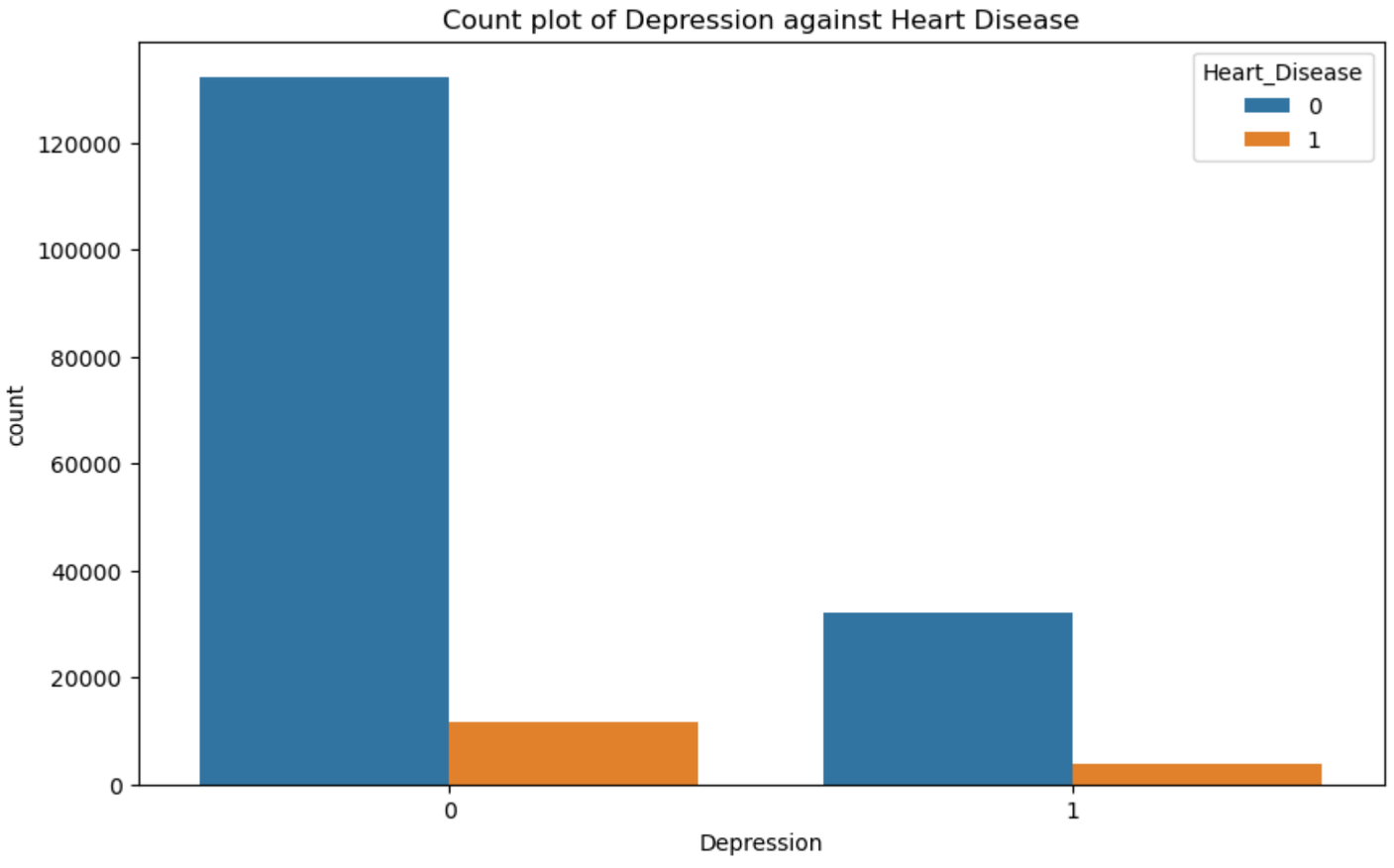
Count plot of General_Health against Heart Disease
Heart_Disease
0
1
count
60000
50000
40000
30000
20000
10000
0
0
1
2
3
4
General_Health

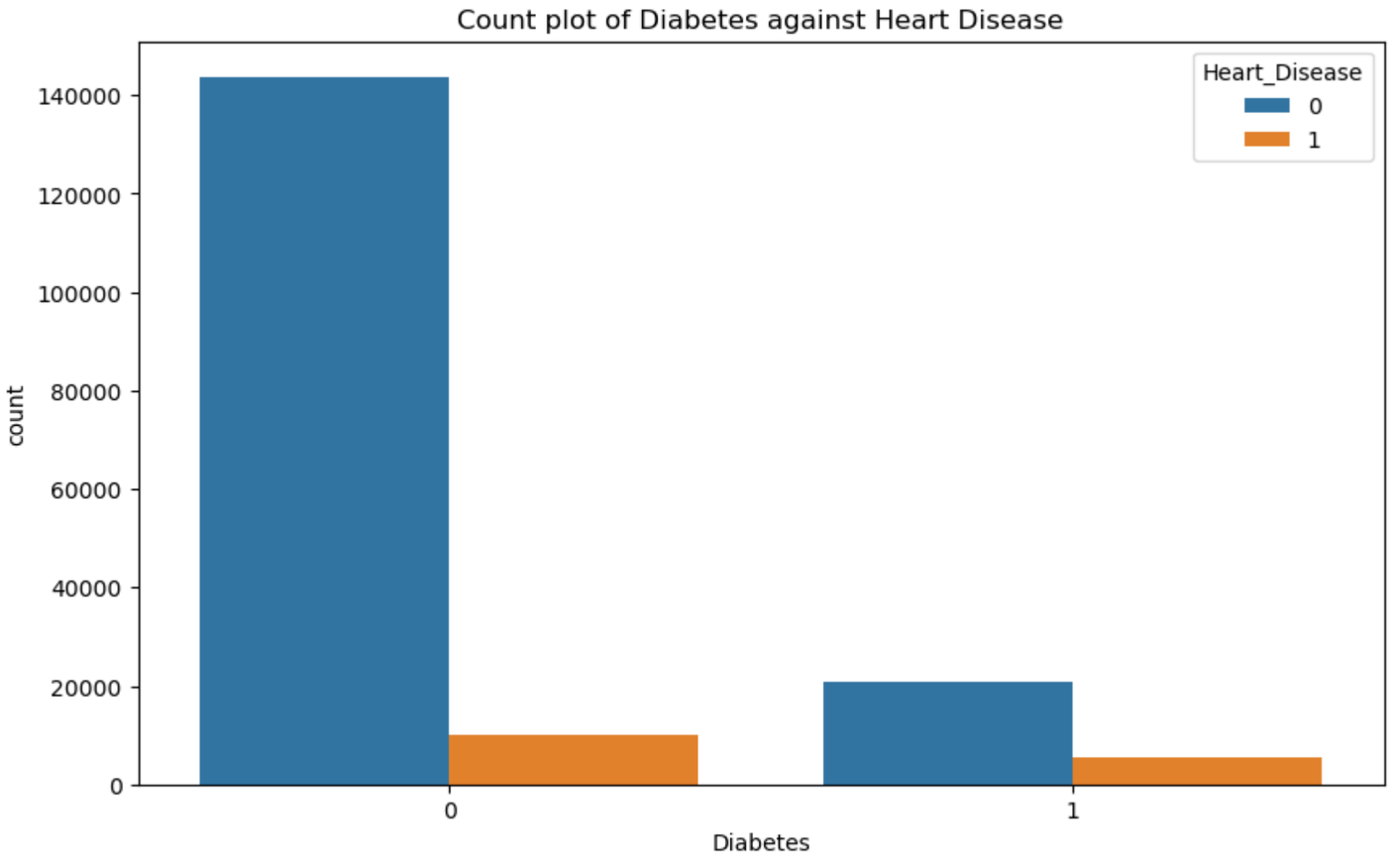

Count plot of Checkup against Heart Disease
Heart_Disease
0
1
count
120000
100000
80000
60000
40000
20000
0
0.0
0.2
1.0
2.0
4.0
Checkup

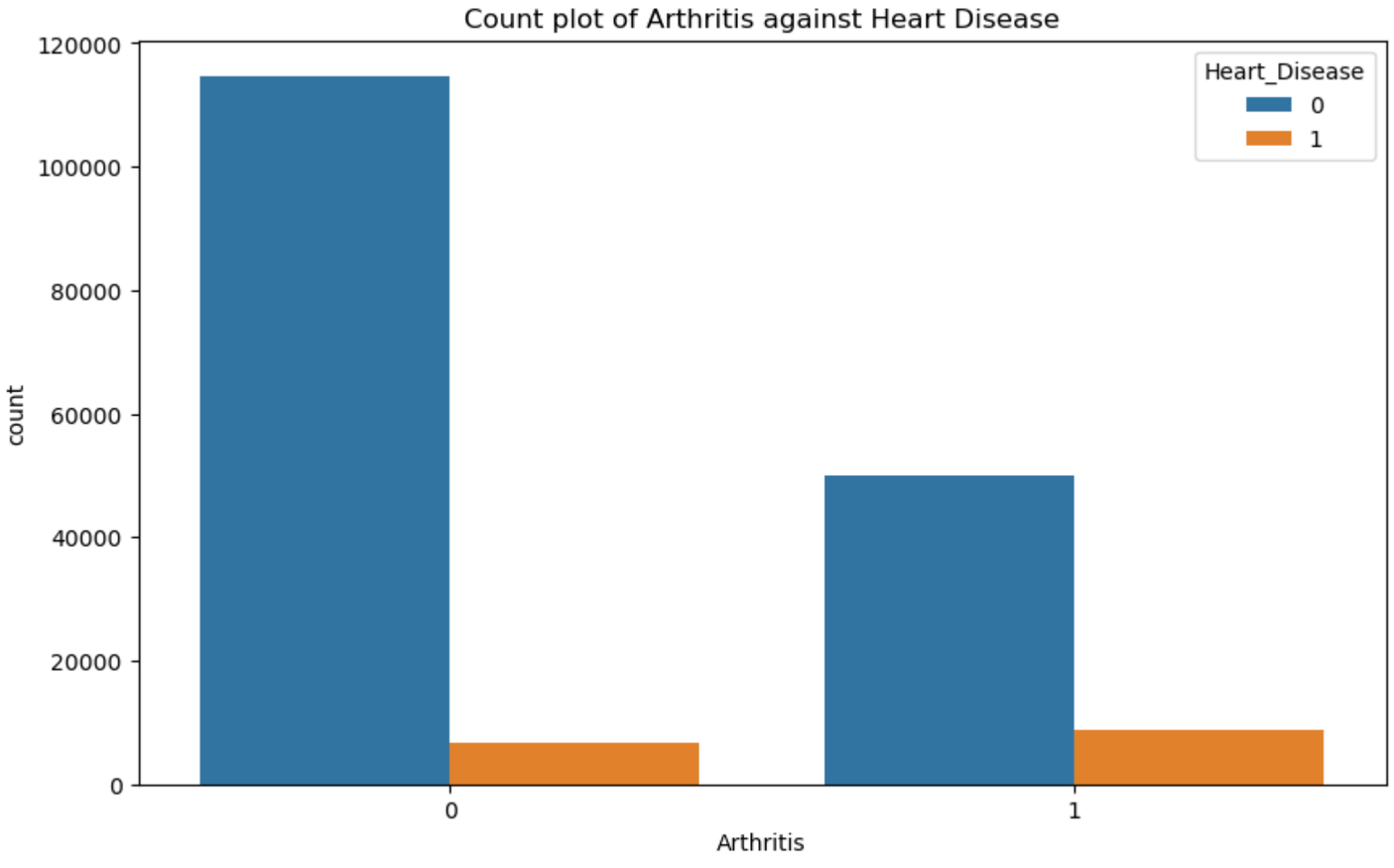

Count plot of Exercise against Heart Disease
Heart_Disease
0
1
count
120000
100000
80000
60000
40000
20000
0
0
1
Exercise

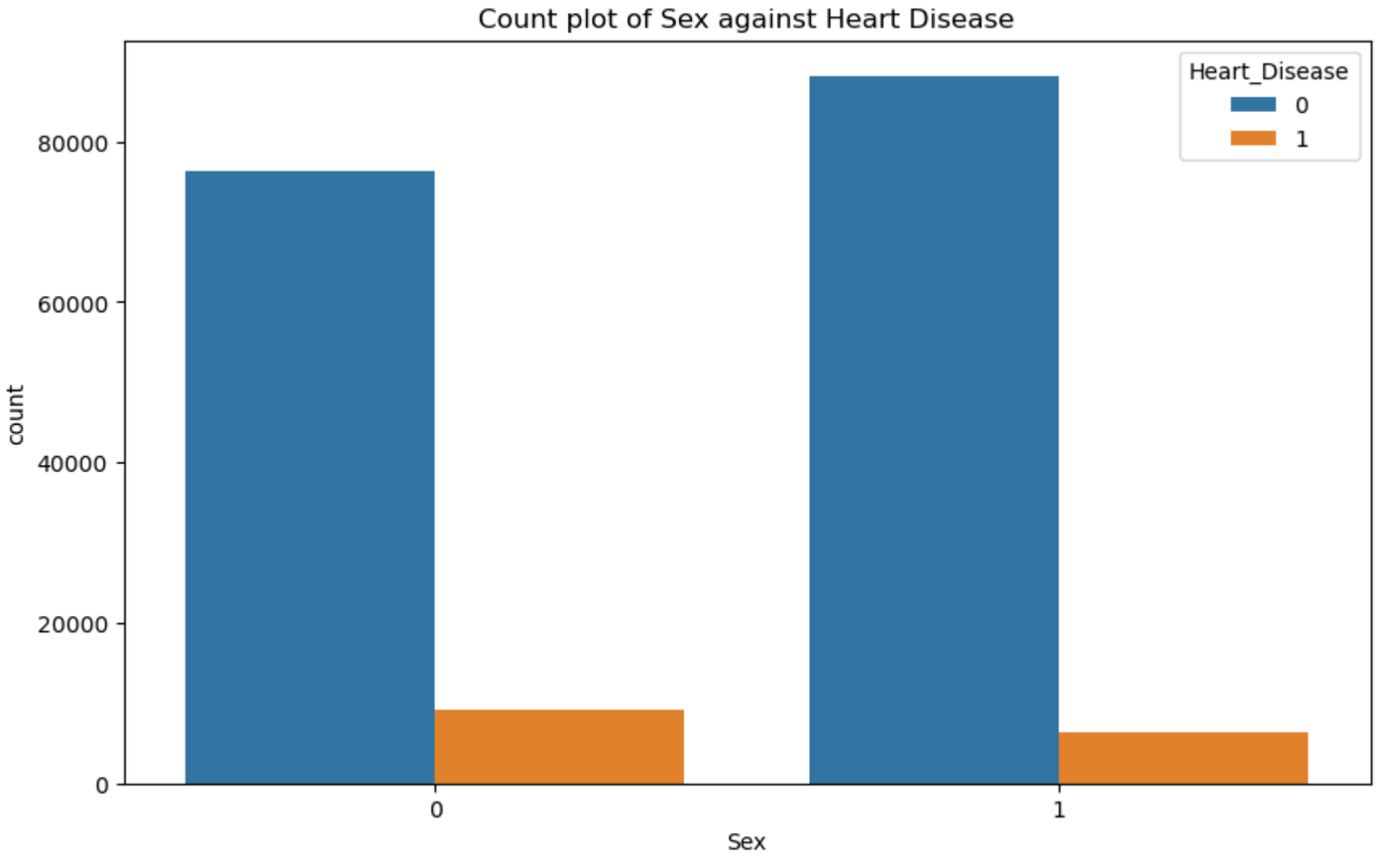

Count plot of Skin_Cancer against Heart Disease
Heart_Disease
0
1
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
Skin_Cancer

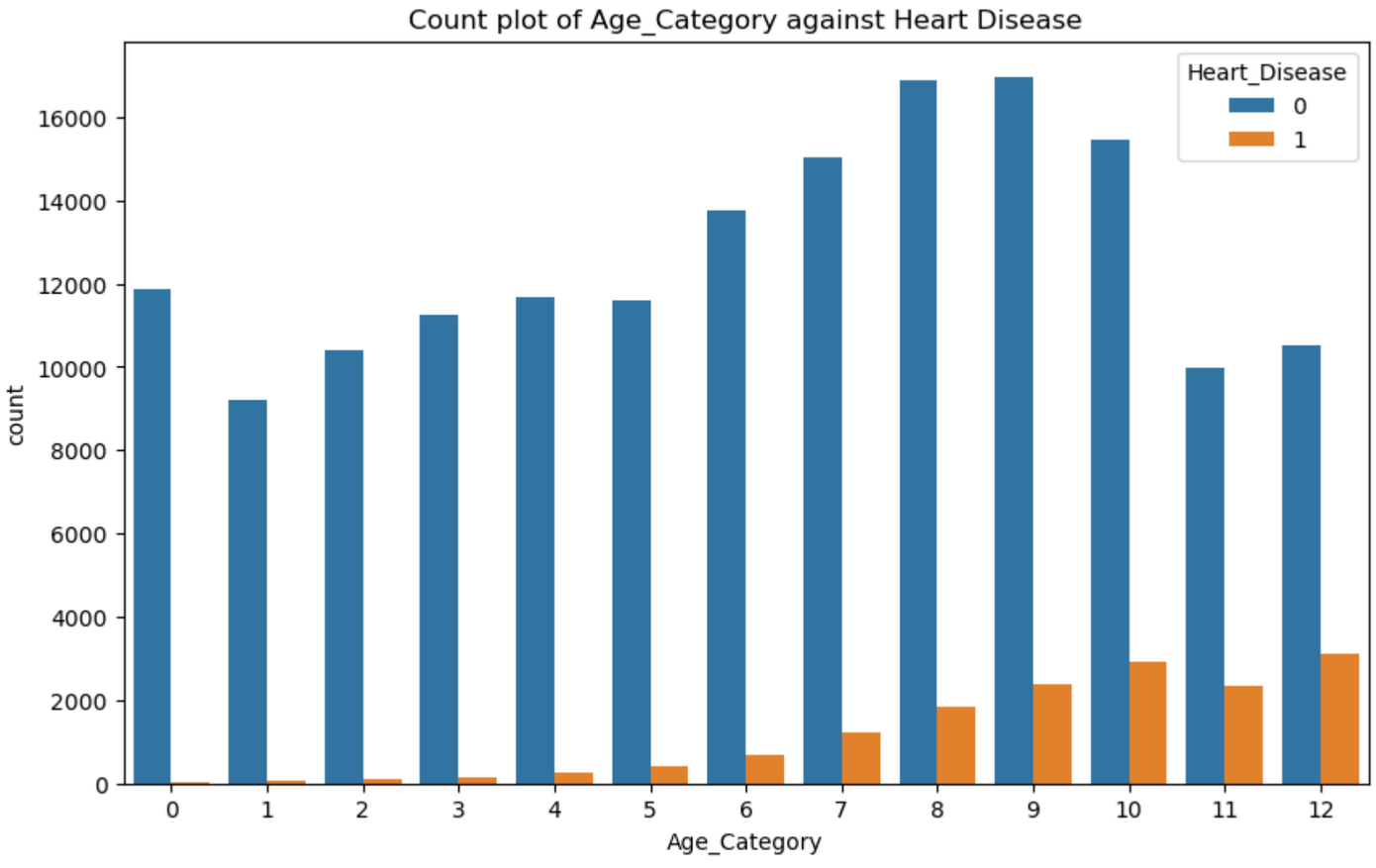

Count plot of Other_Cancer against Heart Disease
Heart_Disease
0
1
count
140000
120000
100000
80000
60000
40000
20000
0
0
1
Other_Cancer

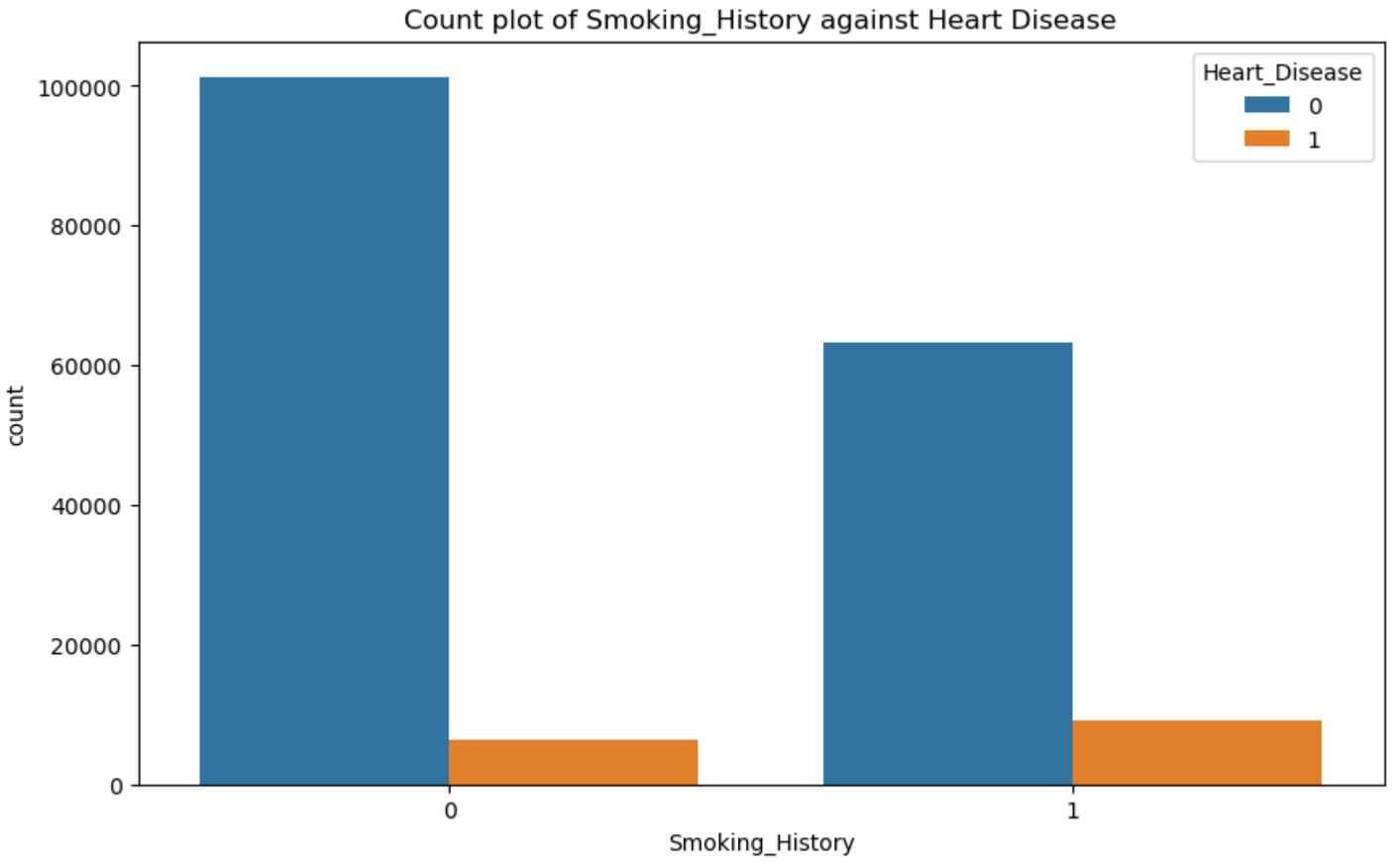

Count plot of Depression against Heart Disease
Heart_Disease
0
1
count
120000
100000
80000
60000
40000
20000
0
0
1
Depression

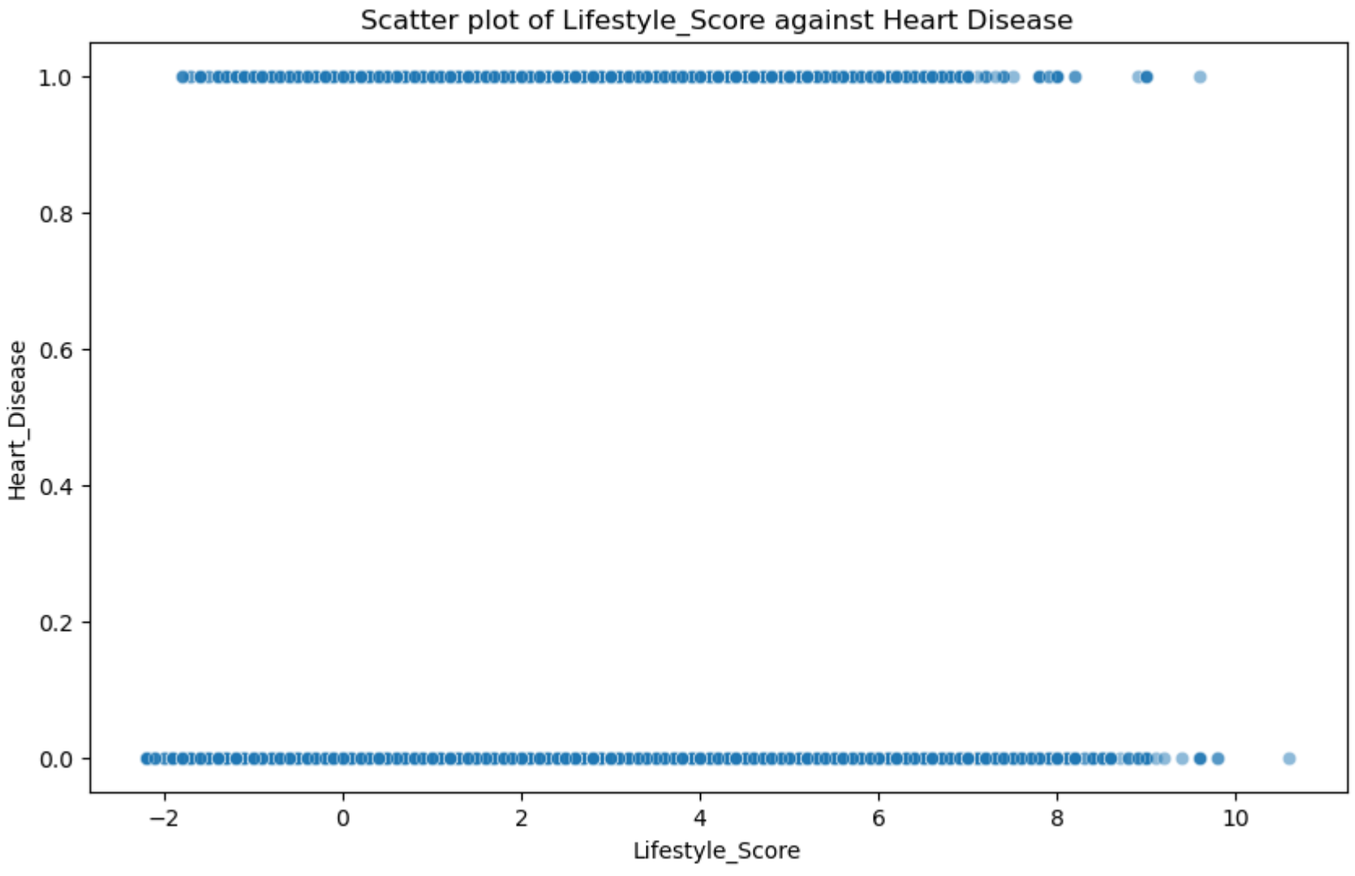

Count plot of Diabetes against Heart Disease
Heart_Disease
0
1
count
140000
120000
100000
80000
60000
40000
20000
0
0
1
Diabetes


Count plot of Arthritis against Heart Disease
Heart_Disease
0
1
count
120000
100000
80000
60000
40000
20000
0
0
1
Arthritis

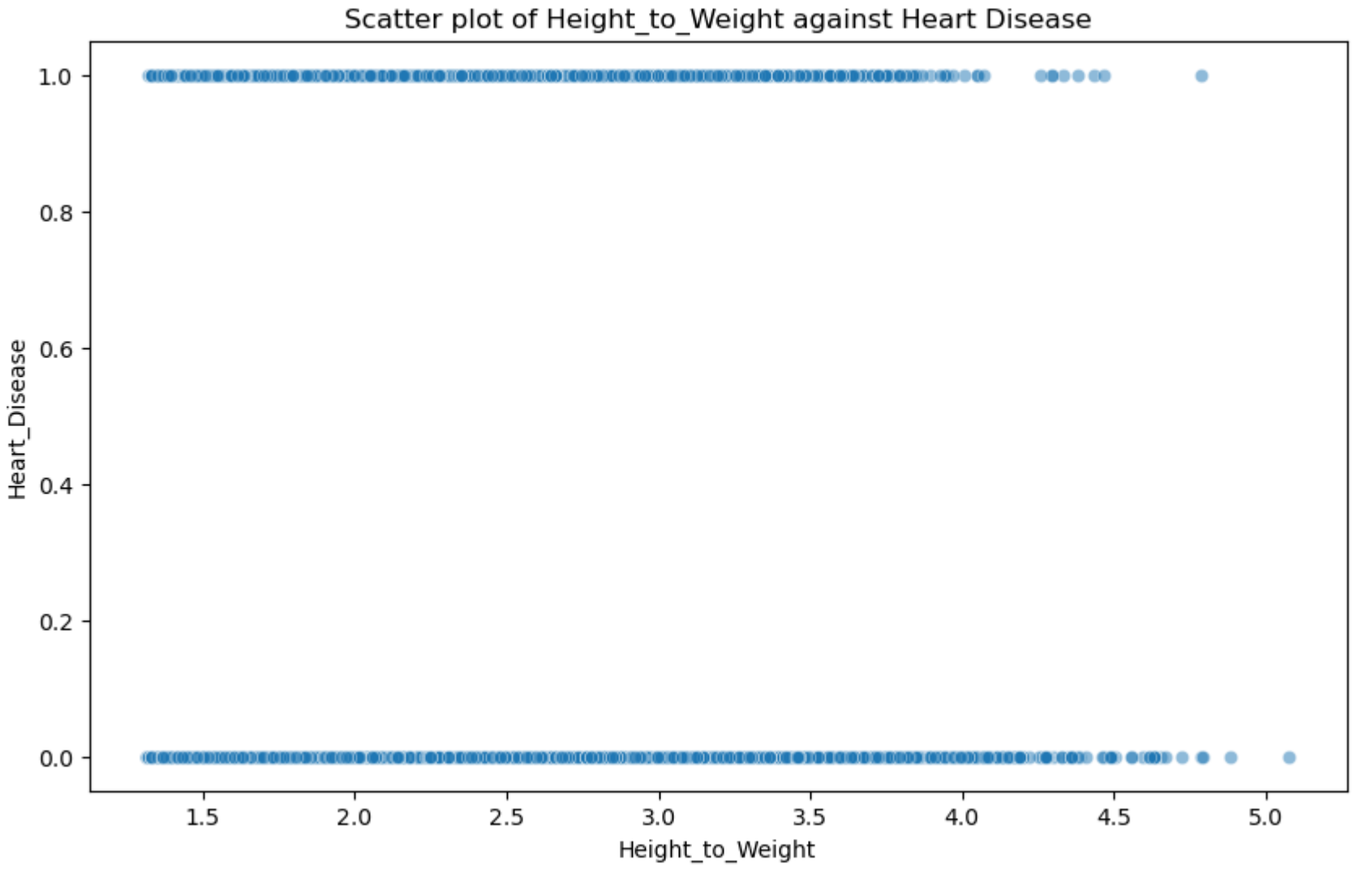

Count plot of Sex against Heart Disease
Heart_Disease
0
1
count
80000
60000
40000
20000
0
0
1
Sex

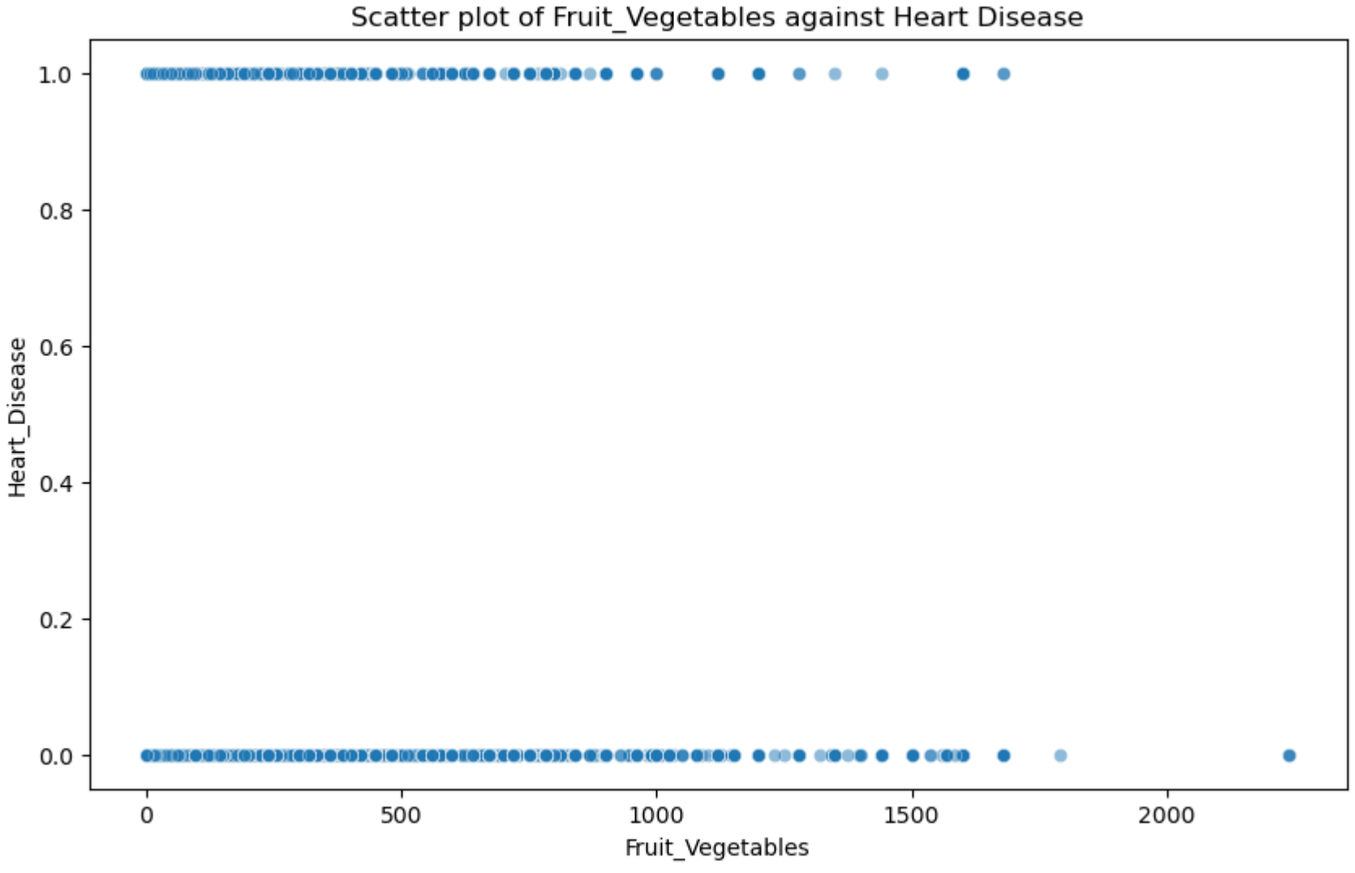

Count plot of Age_Category against Heart Disease
Heart_Disease
0
1
count
16000
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
6
7
8
9
10
11
12
Age_Category

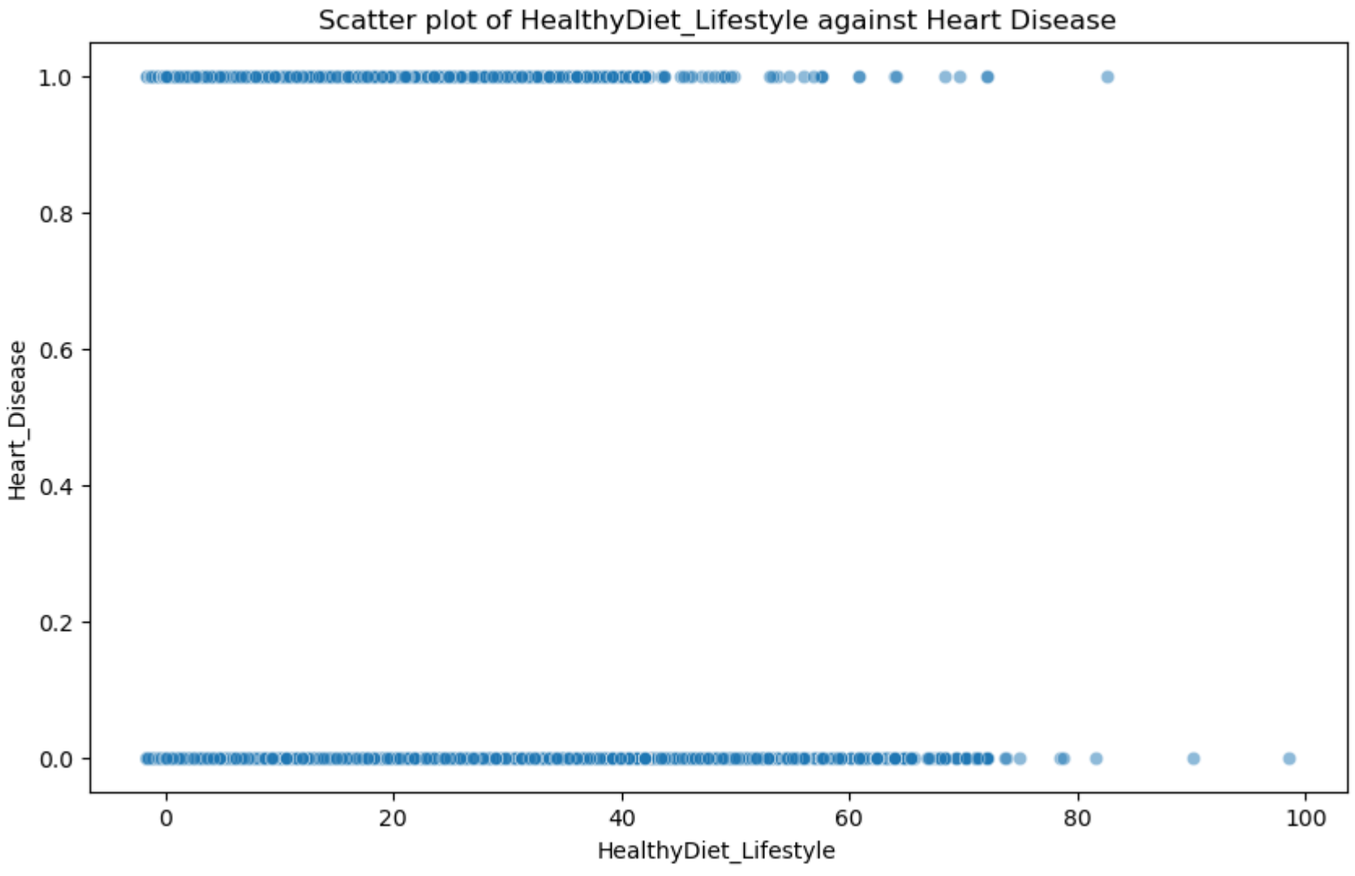

Count plot of Smoking_History against Heart Disease
Heart_Disease
0
1
count
100000
80000
60000
40000
20000
0
0
1
Smoking_History

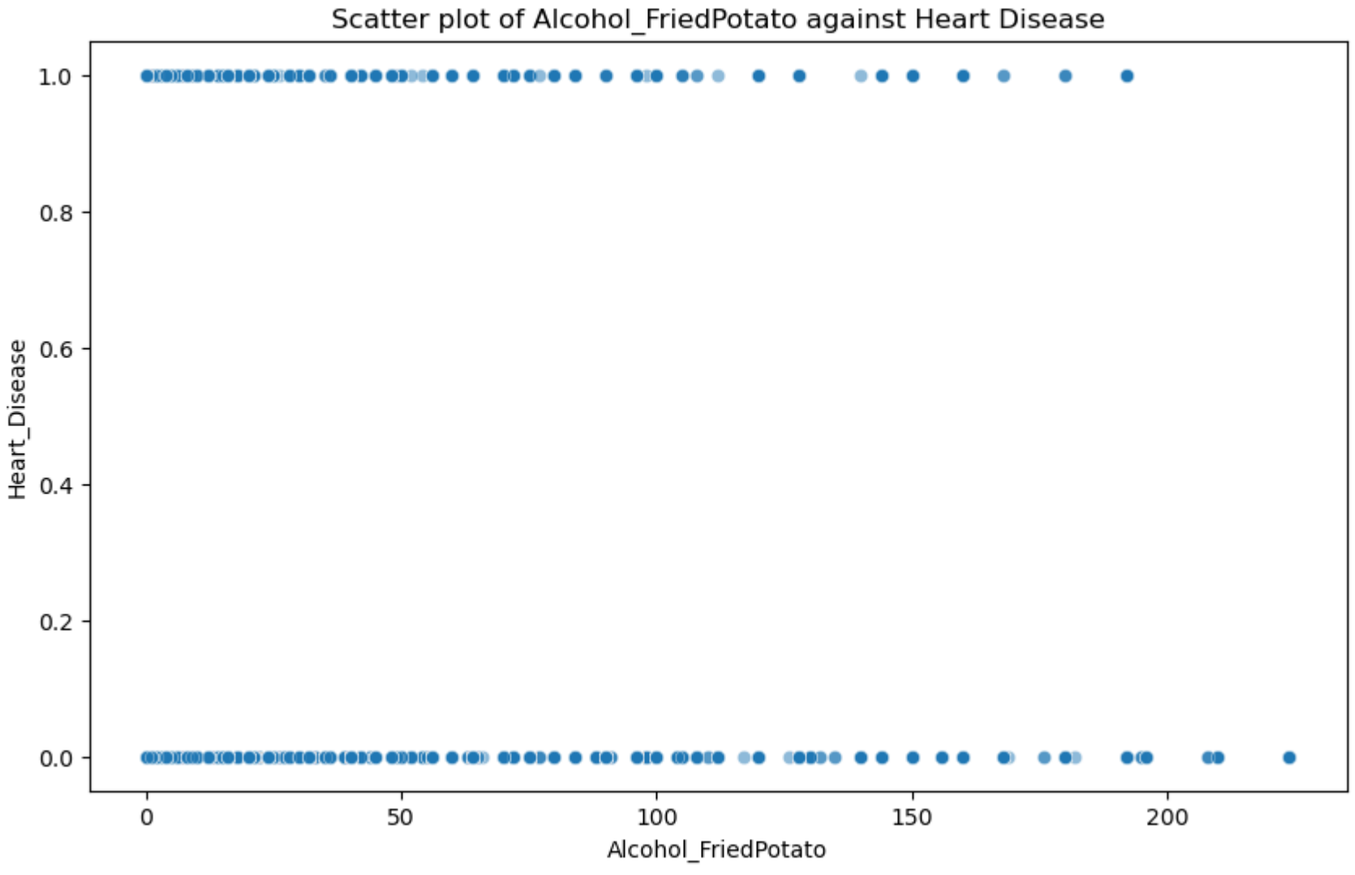

Scatter plot of Lifestyle_Score against Heart Disease
Heart_Disease
1.0
0.8
0.6
0.4
0.2
0.0
−2
0
2
4
6
8
10
Lifestyle_Score

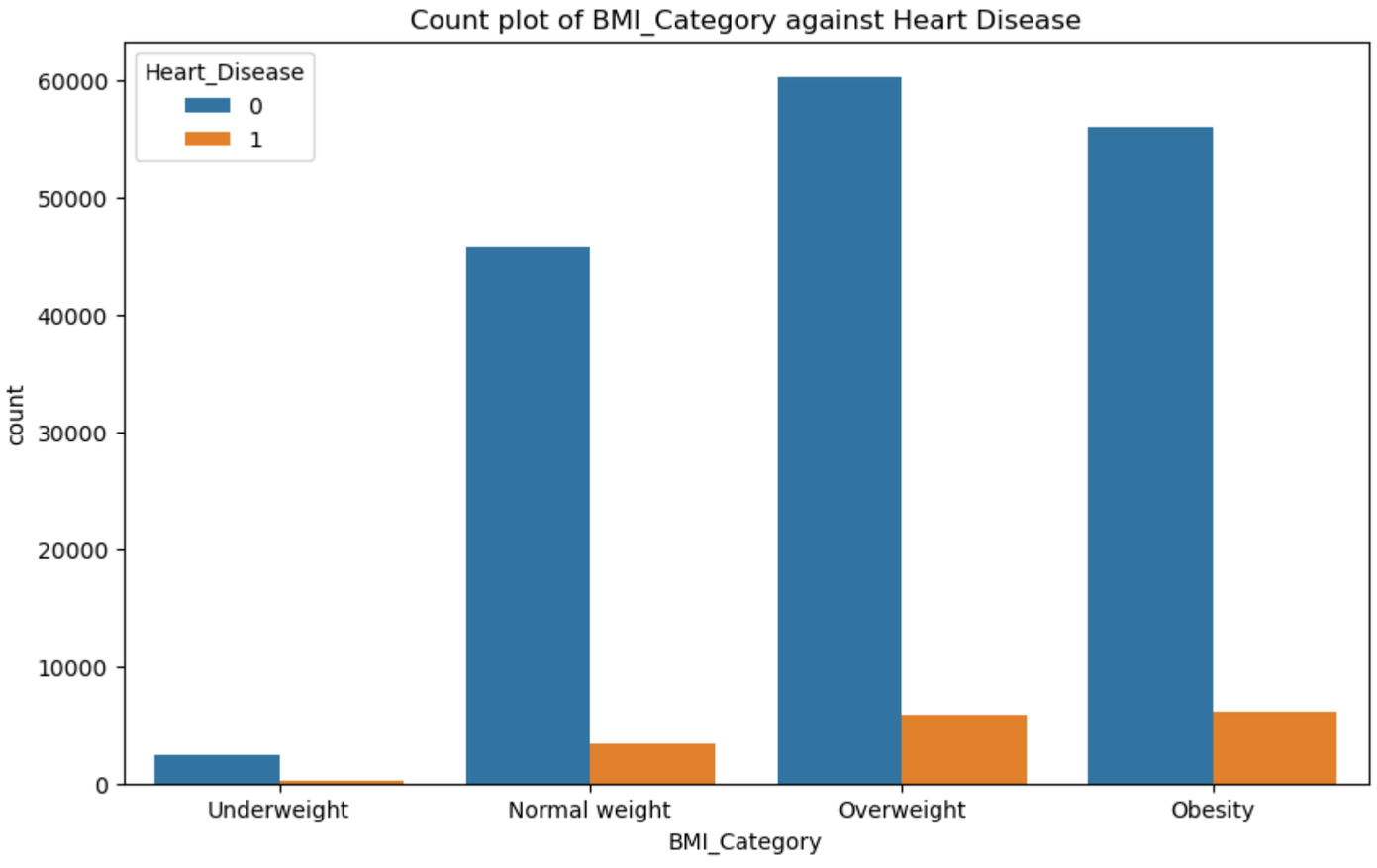
Scatter plot of Healthy_Diet_Score against Heart Disease
Heart_Disease
Healthy_Diet_Score
1.0
0.8
0.6
0.4
0.2
0.0
−2
0
2
4
6
8

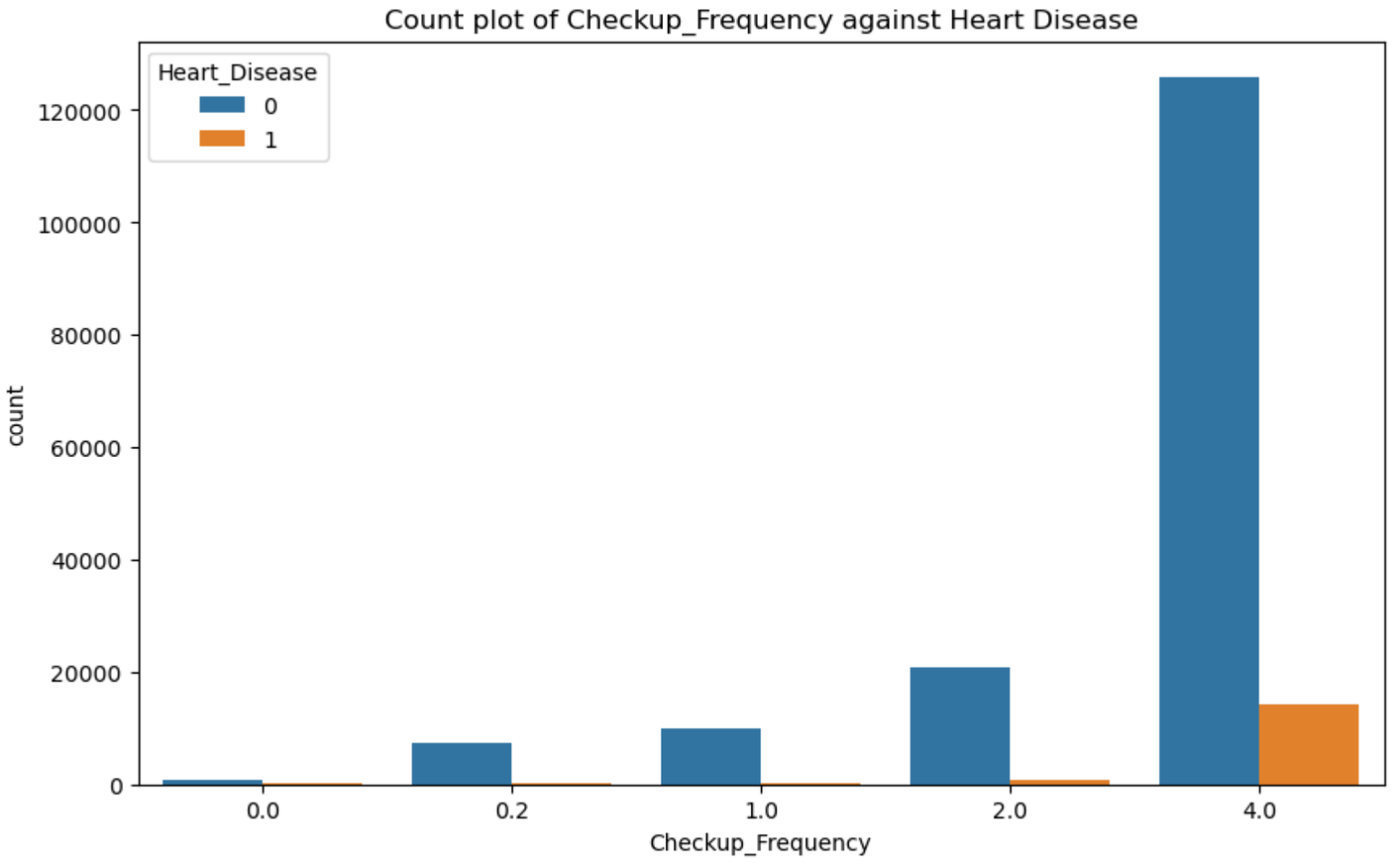

Scatter plot of Height_to_Weight against Heart Disease
Heart_Disease
Height_to_Weight
1.0
0.8
0.6
0.4
0.2
0.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

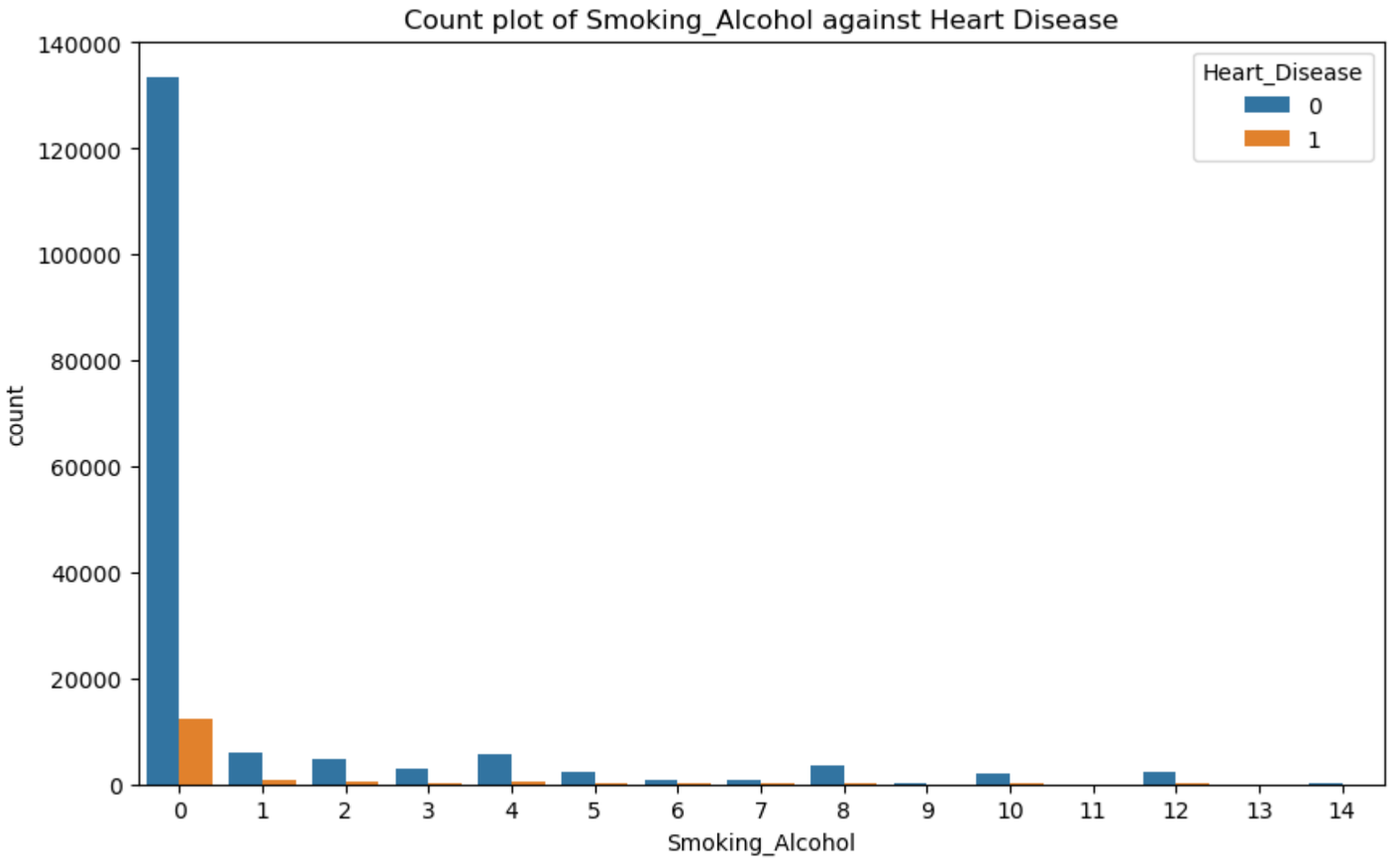

Scatter plot of Fruit_Vegetables against Heart Disease
Heart_Disease
1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
Fruit_Vegetables

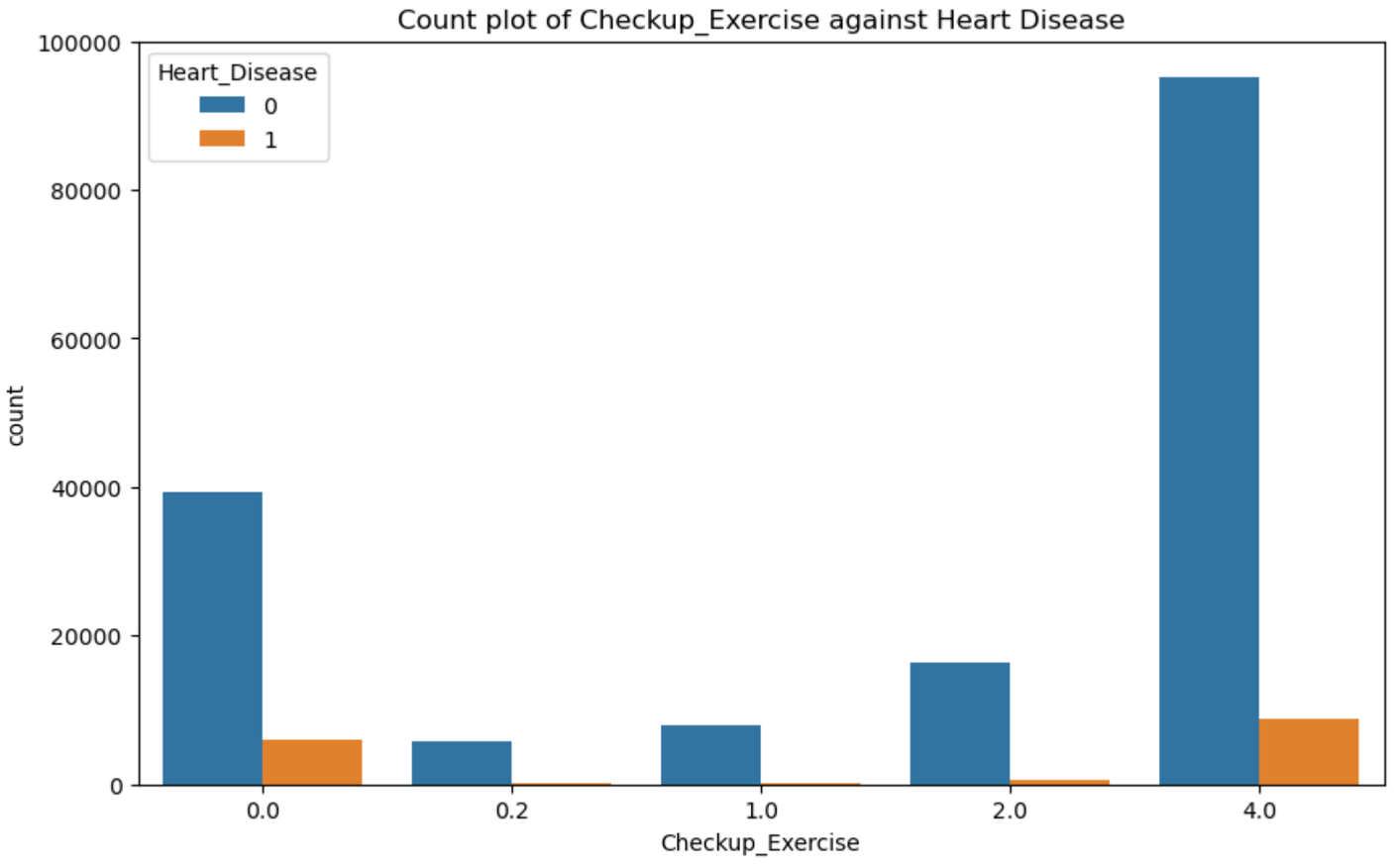

Scatter plot of HealthyDiet_Lifestyle against Heart Disease
Heart_Disease
HealthyDiet_Lifestyle
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100


Scatter plot of Alcohol_FriedPotato against Heart Disease
Heart_Disease
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
Alcohol_FriedPotato


Count plot of BMI_Category against Heart Disease
Heart_Disease
0
1
count
60000
50000
40000
30000
20000
10000
0
Underweight
Normal weight
Overweight
Obesity
BMI_Category


Count plot of Checkup_Frequency against Heart Disease
Heart_Disease
0
1
count
120000
100000
80000
60000
40000
20000
0
0.0
0.2
1.0
2.0
4.0
Checkup_Frequency

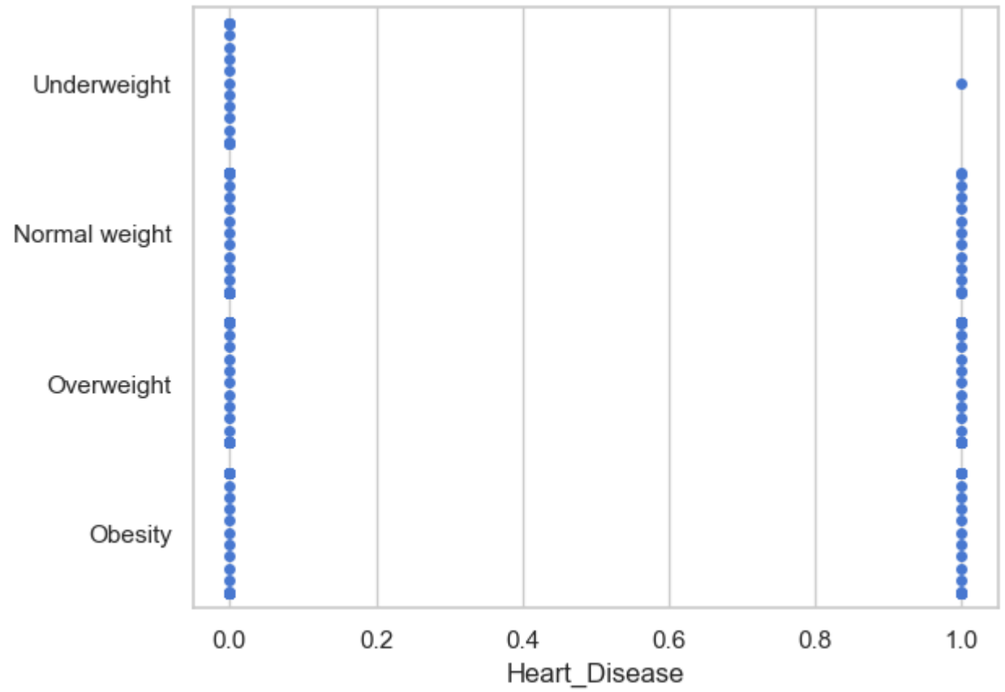

Count plot of Smoking_Alcohol against Heart Disease
Heart_Disease
0
1
count
140000
120000
100000
80000
60000
40000
20000
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Smoking_Alcohol



The above scatterplos and countplots shows the various factors affecting Heart Disease. We can clearly see how people in higher Age_Category and people with low General_Health are more prone to heart diseases.

In [ ]:
```
##---------Type the code below this line-----------------##
import seaborn as sns
sns.set_theme(style="whitegrid", palette="muted")

# Draw a categorical scatterplot to show each observation
ax = sns.swarmplot(data=dataset_sp.sample(1000), x="Heart_Disease", y="BMI_Category")
ax.set(ylabel="")
```

Out[ ]:
```
[Text(-36.375, 0.5, '')]
```



The above scatter plot clearly shows that BMI has a direct effect on heart disease

4.2 EDA using visuals

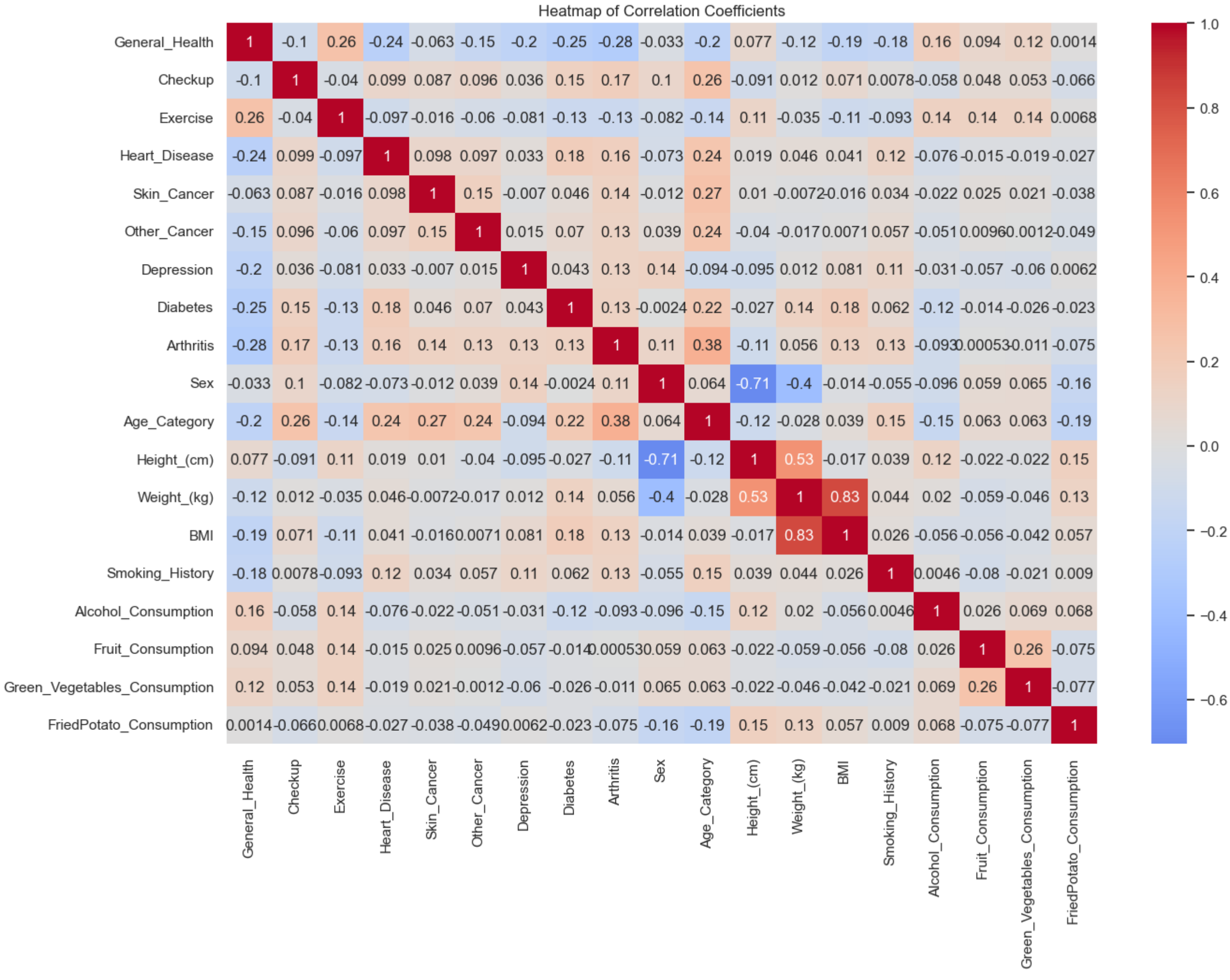
- Use (minimum) 2 plots (pair plot, heat map, correlation plot, regression plot...) to identify the optimal set of attributes that can be used for classification.
- Name them, explain why you think they can be helpful in the task and perform the plot as well. Unless proper justification for the choice of plots given, no credit will be awarded.

Score: 2 Marks

Heat Map of Correlation Coefficients:

Justification: A heatmap displays the correlation coefficients between variables in a visually intuitive manner. If a feature has a high correlation with the target variable, it can be an indication of its potential importance in classification. Additionally, if two features are highly correlated with each other (but not necessarily with the target), one of them might be redundant.

In [ ]:
```
plt.figure(figsize=(15, 10))
correlation_matrix = dataset.corr()  # Calculates the Pearson correlation by default
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", center=0)
plt.title('Heatmap of Correlation Coefficients')
plt.show()
```

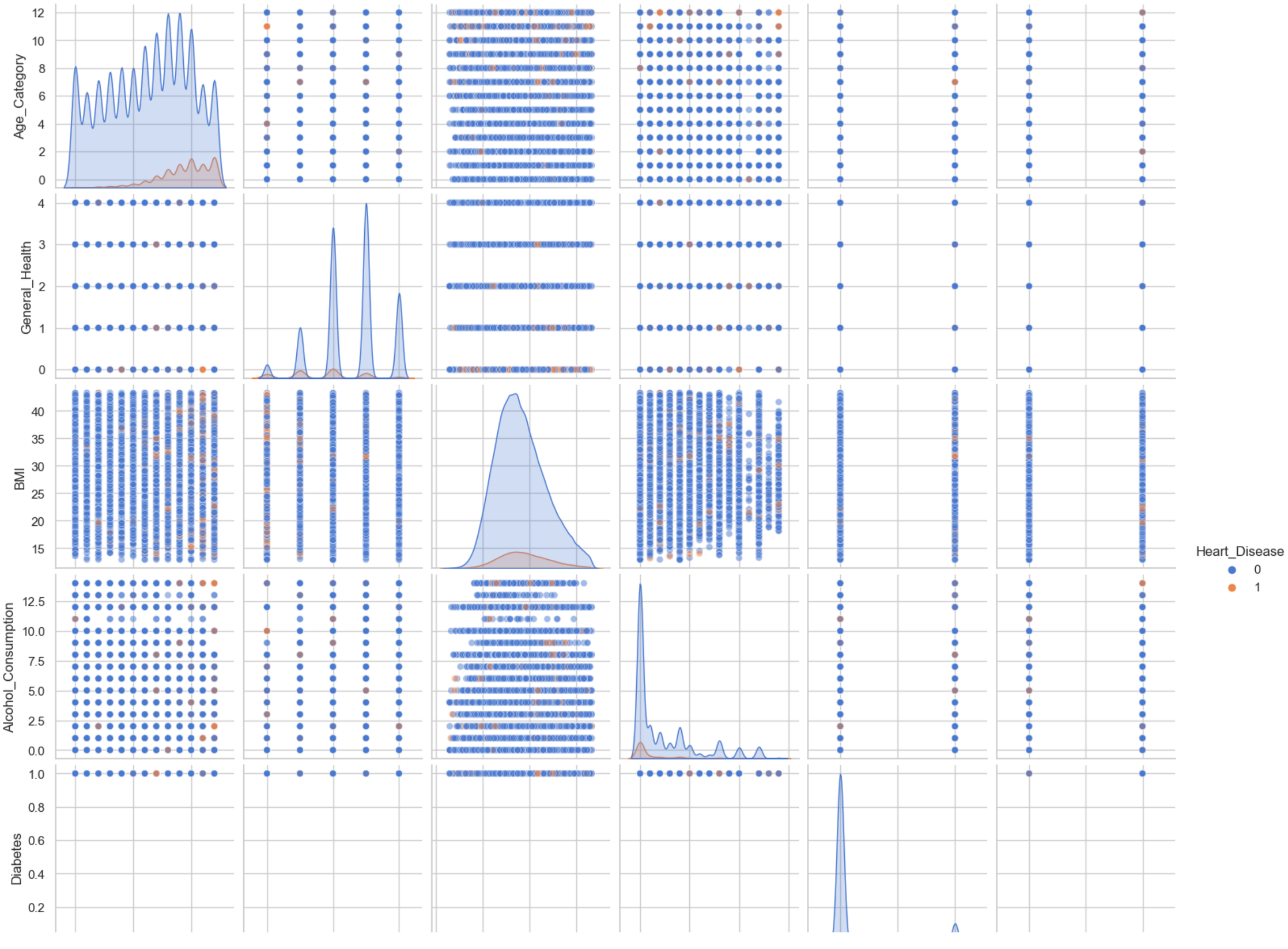

Heatmap of Correlation Coefficients
General_Health 1 -0.1 0.26 -0.24 -0.063 -0.15 -0.2 -0.25 -0.28 -0.033 -0.2 0.077 -0.12 -0.19 -0.18 0.16 0.094 0.12 0.0014
Checkup -0.1 1 -0.04 0.099 0.087 0.096 0.036 0.15 0.17 0.1 0.26 -0.091 0.012 0.071 0.0078 -0.058 0.048 0.053 -0.066
Exercise 0.26 -0.04 1 -0.097 -0.016 -0.06 -0.081 -0.13 -0.13 -0.082 -0.14 0.11 -0.035 -0.11 -0.093 0.14 0.14 0.14 0.0068
Heart_Disease -0.24 0.099 -0.097 1 0.098 0.097 0.033 0.18 0.16 -0.073 0.24 0.019 0.046 0.041 0.12 -0.076 -0.015 -0.019 -0.027
Skin_Cancer -0.063 0.087 -0.016 0.098 1 0.15 -0.007 0.046 0.14 -0.012 0.27 0.01 -0.0072 -0.016 0.034 -0.022 0.025 0.021 -0.038
Other_Cancer -0.15 0.096 -0.06 0.097 0.15 1 0.015 0.07 0.13 0.039 0.24 -0.04 -0.017 0.0071 0.057 -0.051 0.0096 0.0012 -0.049
Depression -0.2 0.036 -0.081 0.033 -0.007 0.015 1 0.043 0.13 0.14 -0.094 -0.095 0.012 0.081 0.11 -0.031 -0.057 -0.06 0.0062
Diabetes -0.25 0.15 -0.13 0.18 0.046 0.07 0.043 1 0.13 -0.0024 0.22 -0.027 0.14 0.18 0.062 -0.12 -0.014 -0.026 -0.023
Arthritis -0.28 0.17 -0.13 0.16 0.14 0.13 0.13 0.13 1 0.11 0.38 -0.11 0.056 0.13 0.13 -0.093 0.00053 -0.011 -0.075
Sex -0.033 0.1 -0.082 -0.073 -0.012 0.039 0.14 -0.0024 0.11 1 0.064 -0.71 -0.4 -0.014 -0.055 -0.096 0.059 0.065 -0.16
Age_Category -0.2 0.26 -0.14 0.24 0.27 0.24 -0.094 0.22 0.38 0.064 1 -0.12 -0.028 0.039 0.15 -0.15 0.063 0.063 -0.19
Height_(cm) 0.077 -0.091 0.11 0.019 0.01 -0.04 -0.095 -0.027 -0.11 -0.71 -0.12 1 0.53 -0.017 0.039 0.12 -0.022 -0.022 0.15
Weight_(kg) -0.12 0.012 -0.035 0.046 -0.0072 -0.017 0.012 0.14 0.056 -0.4 -0.028 0.53 1 0.83 0.044 0.02 -0.059 -0.046 0.13
BMI -0.19 0.071 -0.11 0.041 -0.016 0.0071 0.081 0.18 0.13 -0.014 0.039 -0.017 0.83 1 0.026 -0.056 -0.056 -0.042 0.057
Smoking_History -0.18 0.0078 -0.093 0.12 0.034 0.057 0.11 0.062 0.13 -0.055 0.15 0.039 0.044 0.026 1 0.0046 -0.08 -0.021 0.009
Alcohol_Consumption 0.16 -0.058 0.14 -0.076 -0.022 -0.051 -0.031 -0.12 -0.093 -0.096 -0.15 0.12 0.02 -0.056 0.0046 1 0.026 0.069 0.068
Fruit_Consumption 0.094 0.048 0.14 -0.015 0.025 0.0096 -0.057 -0.014 0.00053 0.059 0.063 -0.022 -0.059 -0.056 -0.08 0.026 1 0.26 -0.075
Green_Vegetables_Consumption 0.12 0.053 0.14 -0.019 0.021 -0.0012 -0.06 -0.026 -0.011 0.065 0.063 -0.022 -0.046 -0.042 -0.021 0.069 0.26 1 -0.077
FriedPotato_Consumption 0.0014 -0.066 0.0068 -0.027 -0.038 -0.049 0.0062 -0.023 -0.075 -0.16 -0.19 0.15 0.13 0.057 0.009 0.068 -0.075 -0.077 1
General_Health
Checkup
Exercise
Heart_Disease
Skin_Cancer
Other_Cancer
Depression
Diabetes
Arthritis
Sex
Age_Category
Height_(cm)
Weight_(kg)
BMI
Smoking_History
Alcohol_Consumption
Fruit_Consumption
Green_Vegetables_Consumption
FriedPotato_Consumption
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

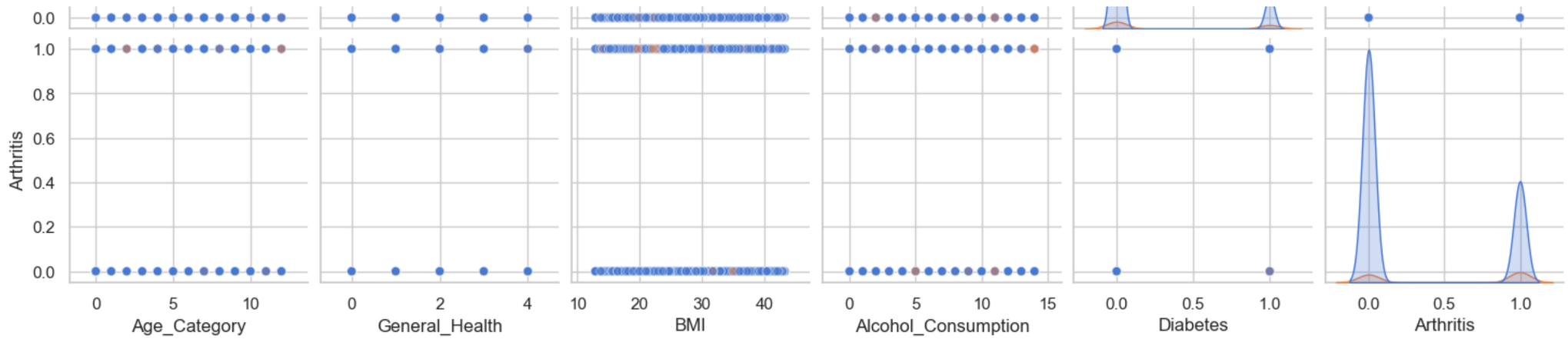
Pair Plot (scatterplot matrix):

Justification: A pair plot allows for visualization of pairwise relationships between multiple variables. For classification, this is useful because you can visually see how different classes are distributed across different feature combinations. If certain feature combinations result in distinct clusters of classes, those features are likely valuable for classification.

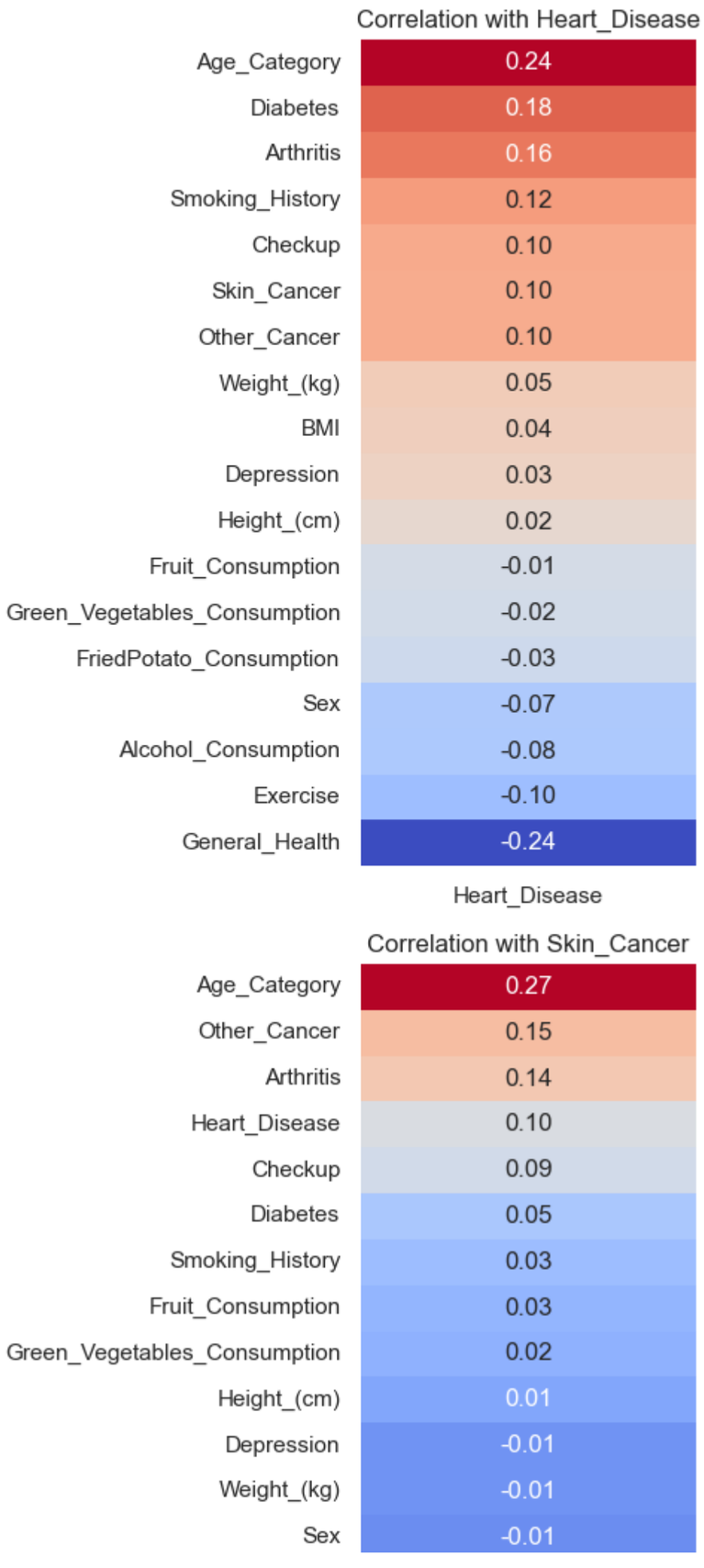
In [ ]:
```
selected_features = ['Age_Category', 'General_Health', 'BMI', 'Alcohol_Consumption', 'Diabetes', 'Arthritis', 'Heart_Disease']  # Top features for illustration
sns.pairplot(dataset[selected_features], hue='Heart_Disease', plot_kws={'alpha': 0.5})
plt.suptitle('Pair Plot of Selected Features', y=1.02)
plt.show()
```

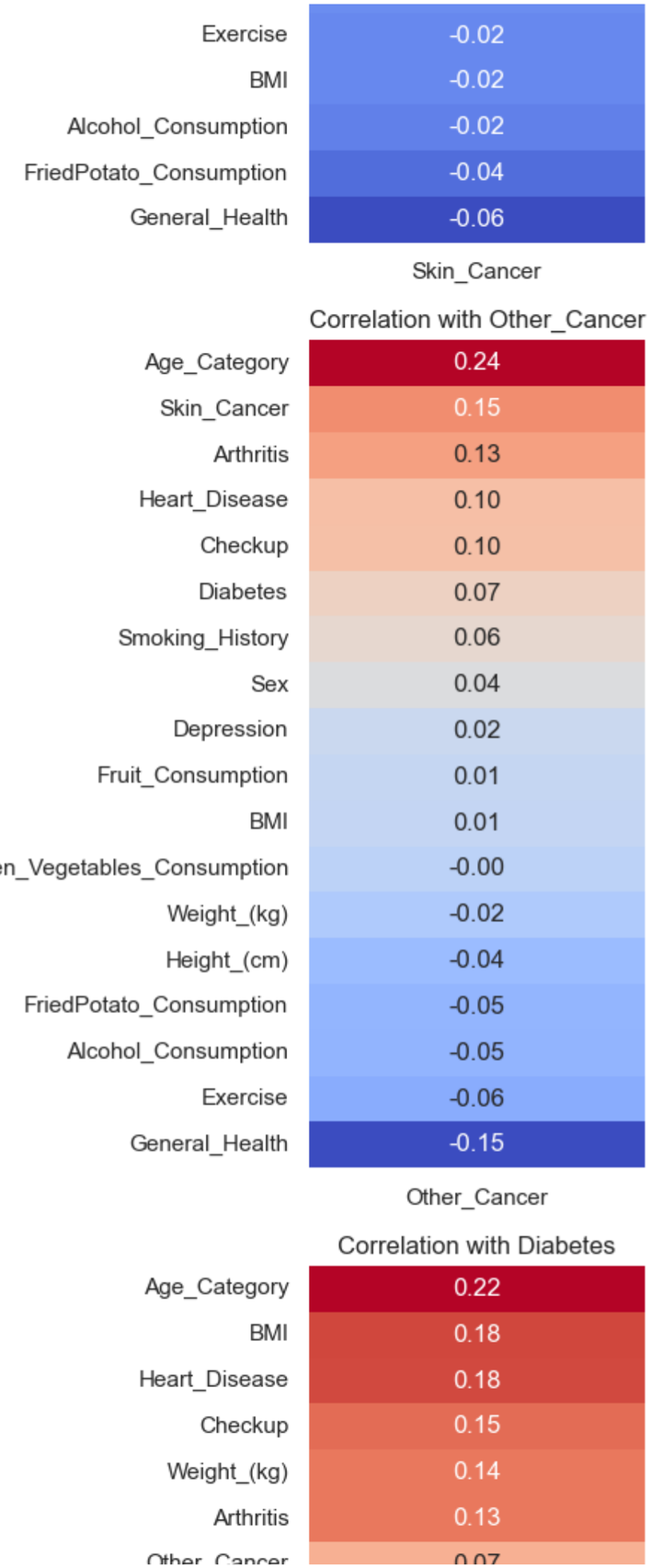
Pair Plot of Selected Features

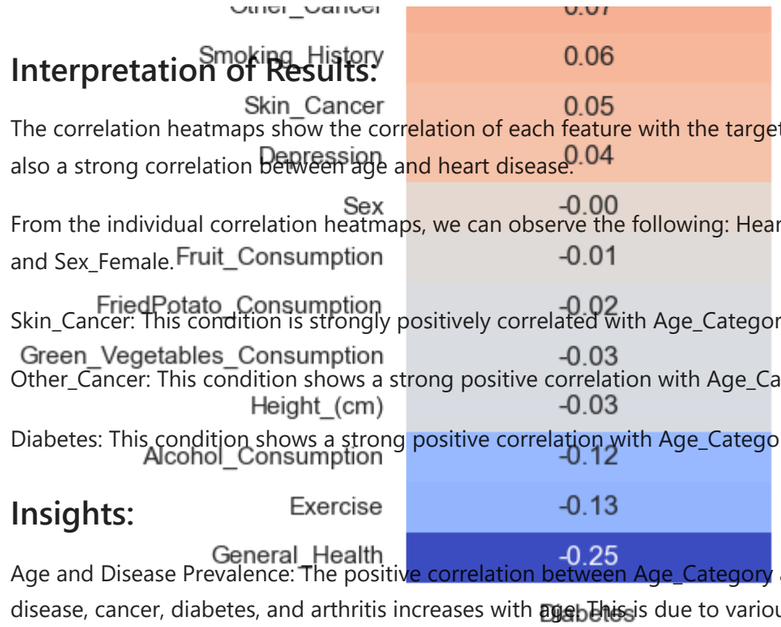




In [ ]:
```
# Compute the correlation of each feature with the disease variables
disease_variables = ['Heart_Disease', 'Skin_Cancer', 'Other_Cancer', 'Diabetes']

# Compute the correlation matrix
corr = dataset.corr()

# Create a subplot for each disease
fig, axs = plt.subplots(len(disease_variables), 1, figsize=(5, 25))

for ax, disease in zip(axs, disease_variables):
    # Compute the correlation with the disease
    target_corr = corr[disease].drop(disease)

    # Sort correlation values in descending order
    target_corr_sorted = target_corr.sort_values(ascending=False)

    # Plot a heatmap of the correlations with the disease
    sns.heatmap(target_corr_sorted.to_frame(), cmap="coolwarm", annot=True, fmt='.2f', cbar=False, ax=ax)
    ax.set_title('Correlation with ' + disease)

plt.tight_layout()
plt.show()
```


Correlation with Heart_Disease
Age_Category 0.24
Diabetes 0.18
Arthritis 0.16
Smoking_History 0.12
Checkup 0.10
Skin_Cancer 0.10
Other_Cancer 0.10
Weight_(kg) 0.05
BMI 0.04
Depression 0.03
Height_(cm) 0.02
Fruit_Consumption -0.01
Green_Vegetables_Consumption -0.02
FriedPotato_Consumption -0.03
Sex -0.07
Alcohol_Consumption -0.08
Exercise -0.10
General_Health -0.24
Heart_Disease
Correlation with Skin_Cancer
Age_Category 0.27
Other_Cancer 0.15
Arthritis 0.14
Heart_Disease 0.10
Checkup 0.09
Diabetes 0.05
Smoking_History 0.03
Fruit_Consumption 0.03
Green_Vegetables_Consumption 0.02
Height_(cm) 0.01
Depression -0.01
Weight_(kg) -0.01
Sex -0.01


Exercise -0.02
BMI -0.02
Alcohol_Consumption -0.02
FriedPotato_Consumption -0.04
General_Health -0.06
Skin_Cancer
Correlation with Other_Cancer
Age_Category 0.24
Skin_Cancer 0.15
Arthritis 0.13
Heart_Disease 0.10
Checkup 0.10
Diabetes 0.07
Smoking_History 0.06
Sex 0.04
Depression 0.02
Fruit_Consumption 0.01
BMI 0.01
Green_Vegetables_Consumption -0.00
Weight_(kg) -0.02
Height_(cm) -0.04
FriedPotato_Consumption -0.05
Alcohol_Consumption -0.05
Exercise -0.06
General_Health -0.15
Other_Cancer
Correlation with Diabetes
Age_Category 0.22
BMI 0.18
Heart_Disease 0.18
Checkup 0.15
Weight_(kg) 0.14
Arthritis 0.13
Other_Cancer 0.07



Interpretation of Results:

The correlation heatmaps show the correlation of each feature with the target variable: Heart_Disease. We can see that ususally people with heart disease tend to have other diseases like cancer and diabetes too. There is also a strong correlation between age and heart disease.

From the individual correlation heatmaps, we can observe the following: Heart_Disease: This condition shows a strong positive correlation with Age_Category and General_Health, and a negative correlation with Exercise and Sex_Female.

Skin_Cancer: This condition is strongly positively correlated with Age_Category and Sex_Male, and negatively correlated with Sex_Female.

Other_Cancer: This condition shows a strong positive correlation with Age_Category and General_Health, and a negative correlation with Sex_Female.

Diabetes: This condition shows a strong positive correlation with Age_Category, General_Health, and BMI, and a negative correlation with Exercise.

Insights:

Age and Disease Prevalence: The positive correlation between Age_Category and the diseases under study aligns with established medical knowledge. It is well known that the risk of chronic conditions such as heart disease, cancer, diabetes, and arthritis increases with age. This is due to various factors including the cumulative effect of exposure to risk factors, increased wear and tear on the body, and changes in the body's physiological functions.

Health Perception and Disease Prevalence: The negative correlation between self-rated General_Health and disease conditions underlines the importance of patients' perception of their own health. Patients who perceive their health as "Poor" or "Fair" are more likely to have chronic conditions. This could be because the symptoms or management of these conditions impact their perceived health status.

Exercise and Health: The negative correlation between Exercise and diseases such as heart disease, diabetes, and arthritis reaffirms the well-established belief in the health benefits of regular physical activity. Regular exercise can help control weight, reduce the risk of heart diseases, and manage blood sugar and insulin levels, among other benefits.

BMI and Diabetes: The positive correlation between BMI and diabetes aligns with existing knowledge. High BMI, especially obesity, is a known risk factor for type 2 diabetes. Excess fat, particularly if stored around the abdomen, can increase the body's resistance to insulin, leading to increased blood sugar.

Gender and Disease Prevalence: The correlations between Sex and certain diseases reveal interesting patterns. For instance, heart disease is more common in males, which agrees with many studies showing men are at a higher risk of heart disease. Skin cancer, however, is more common in females, which could be due to factors like longer life expectancy or different exposure to risk factors.

It's important to note that correlation does not imply causation. While these correlations provide valuable insights, they don't tell us whether one variable causes or directly influences another. Further statistical or experimental studies would be needed to determine causal relationships.

5. Data Wrangling

5.1 Univariate Filters

Numerical and Categorical Data

- Identify top 5 significant features by evaluating each feature independently with respect to the target variable by exploring

1. Mutual Information (Information Gain)
2. Gini index
3. Gain Ratio
4. Chi-Squared test
5. Fisher Score (From the above 5 you are required to use only any **two**)

For Text data

1. Stemming / Lemmatization.
2. Forming n-grams and storing them in the document vector.
3. TF-IDF (From the above 2 you are required to use only any **two**)

Score: 3 Marks

In [ ]:
```
from sklearn.feature_selection import mutual_info_classif

mi_scores = mutual_info_classif(X, y)
mi_series = pd.Series(mi_scores, index=X.columns)

mi_top5 = mi_series.sort_values(ascending=False).head(5)
print("Top 5 features using Mutual Information:", mi_top5.index.tolist())
```

```
Top 5 features using Mutual Information: ['Exercise', 'General_Health', 'Age_Category', 'Sex', 'Checkup']
```

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(criterion='gini')
clf.fit(X, y)

gini_importances = clf.feature_importances_
gini_series = pd.Series(gini_importances, index=X.columns)

gini_top5 = gini_series.sort_values(ascending=False).head(5)
print("Top 5 features using Gini Index:", gini_top5.index.tolist())
```

```
Top 5 features using Gini Index: ['BMI', 'Weight_(kg)', 'Green_Vegetables_Consumption', 'FriedPotato_Consumption', 'Age_Category']
```

In [ ]:
```
from sklearn.feature_selection import chi2

chi2_scores, _ = chi2(X, y)
chi2_series = pd.Series(chi2_scores, index=X.columns)

chi2_top5 = chi2_series.sort_values(ascending=False).head(5)
print("Top 5 features using Chi-Squared Test:", chi2_top5.index.tolist())
```

```
Top 5 features using Chi-Squared Test: ['Age_Category', 'Alcohol_Consumption', 'Diabetes', 'General_Health', 'Arthritis']
```

In [ ]:
```
from sklearn.feature_selection import f_classif

f_scores, _ = f_classif(X, y)
f_series = pd.Series(f_scores, index=X.columns)

f_top5 = f_series.sort_values(ascending=False).head(5)
print("Top 5 features using Fisher Score:", f_top5.index.tolist())
```

```
Top 5 features using Fisher Score: ['Age_Category', 'General_Health', 'Diabetes', 'Arthritis', 'Smoking_History']
```

5.2 Report observations

Write your observations from the results of each method. Clearly justify your choice of the method.

Score 1 mark

Observations:

1. **Mutual Information**:

- The features identified through Mutual Information largely point towards general health behavior and demographics, with 'Exercise', 'General_Health', and 'Checkup' being indicative of a person's health consciousness and habits. The inclusion of 'Sex' and 'Age_Category' underscores that demographic factors play a significant role in influencing cardiovascular risk.

2. **Gini Index**:
   - The features highlighted by the Gini Index emphasize more on physical metrics and dietary habits. 'BMI' and 'Weight_(kg)' are direct indicators of a person's physical health and are known risk factors for various diseases. The consumption metrics — 'Green_Vegetables_Consumption', 'Fruit_Consumption', and 'FriedPotato_Consumption' — reflect a person's diet, which is crucial for heart health.
3. **Chi-Squared Test**:
   - The Chi-Squared Test brings forth a mix of demographic factors ('Age*Category'), physical health metrics ('Weight*(kg)'), and specific health conditions ('Diabetes' and 'Arthritis'). The presence of 'Diabetes' as a top feature is expected, given its strong association with cardiovascular diseases. 'Alcohol_Consumption' indicates lifestyle choices' impact on heart health.
4. **Fisher Score**:
   - Similar to the Chi-Squared Test, the Fisher Score too emphasizes the importance of 'Age_Category', 'Diabetes', and 'Arthritis'. Additionally, 'Smoking_History' and 'Exercise' further bring forth the importance of lifestyle choices in influencing cardiovascular health.

Overall Justification:

- The varying sets of important features across the methods underscore the importance of considering multiple feature selection approaches. Each method captures a different aspect of the relationship between features and the target.
- The **Mutual Information** method provides insights into general health habits and demographic factors, suggesting that daily habits and age/sex play a crucial role in heart health.
- The **Gini Index** focuses more on dietary habits and physical health metrics. It's beneficial to understand which tangible metrics (like BMI) and dietary habits are linked with heart health.
- Both the **Chi-Squared Test** and **Fisher Score** provide a balanced view, highlighting both specific health conditions and lifestyle choices.

Choice of Method:

- If the goal is to identify modifiable lifestyle habits linked with cardiovascular health, the Mutual Information or Gini Index would be preferred.
- If the focus is on broader risk factors, including existing health conditions and habits, the Chi-Squared Test or Fisher Score would be more appropriate.
- However, a holistic approach might involve taking insights from all methods and potentially combining them to get a comprehensive list of features. For instance, if a feature is identified as important across multiple methods (like 'Age_Category'), it strengthens the case for its significance.

While these methods provide statistical and data-driven insights, it's crucial to integrate domain knowledge and expertise, especially in critical fields like healthcare.

6. Implement Machine Learning Techniques

Use any 2 ML algorithms

1. Classification -- Decision Tree classifier
2. Clustering -- kmeans
3. Association Analysis
4. Anomaly detection
5. Textual data -- Naive Bayes classifier (not taught in this course)

A clear justification have to be given for why a certain algorithm was chosen to address your problem.

Score: 4 Marks (2 marks each for each algorithm)

6.1 ML technique 1 + Justification

Decision Tree Classifier

In [ ]:
```
##---------Type the code below this line-----------------##
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# Assuming 'Heart_Disease' is the target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

predictions = clf.predict(X_test)
print(predictions)
```

```
[0 0 0 ... 0 0 0]
```

Justification

Interpretability: One of the main advantages of decision trees is their inherent interpretability. Given that health-related decisions often require clear rationale and understanding, decision trees provide a visual and logical flow of decisions, making it easier for stakeholders (e.g., doctors, patients) to comprehend the model's choices.

Non-linearity: Decision trees can capture non-linear relationships in the data without requiring transformations. Given the multifaceted nature of health factors, where the interplay between features might not always be linear, this attribute is highly beneficial.

Ability to Handle Mixed Data: Your dataset contains both categorical and numerical data. Decision trees can seamlessly handle this mix without the need for extensive preprocessing.

Feature Importance: Decision trees can rank features based on their importance in making decisions. This can be instrumental in understanding which factors play the most significant role in determining cardiovascular risks.

6.2 ML technique 2 + Justification

Association Analysis

In [ ]:
```
import numpy as np
from mlxtend.frequent_patterns import apriori, association_rules

# Discretize continuous columns like BMI, Alcohol_Consumption, etc., and then one-hot encode
X['BMI_binned'] = pd.cut(X['BMI'], bins=[0.0, 18.5, 24.9, 29.9, np.inf], labels=['Underweight', 'Normal', 'Overweight', 'Obese'])

X['Alcohol_Consumption_binned'] = pd.cut(X['Alcohol_Consumption'], bins=[-1, 10, 20, np.inf], labels=['Low', 'Moderate', 'High'])

X['Fruit_Consumption_binned'] = pd.cut(X['Fruit_Consumption'], bins=[-1, 30, 60, 90, np.inf], labels=['Low', 'Moderate', 'High', 'Very High'])

X['Green_Vegetables_Consumption_binned'] = pd.cut(X['Green_Vegetables_Consumption'], bins=[-1, 30, 60, 90, np.inf], labels=['Low', 'Moderate', 'High', 'Very High'])

X['FriedPotato_Consumption_binned'] = pd.cut(X['FriedPotato_Consumption'], bins=[-1, 30, 60, 90, np.inf], labels=['Low', 'Moderate', 'High', 'Very High'])

# Then one-hot encode
df_bmi_encoded = pd.get_dummies(X['BMI_binned'], prefix='BMI')
df_alcohol_encoded = pd.get_dummies(X['Alcohol_Consumption_binned'], prefix='Alcohol')
df_fruit_encoded = pd.get_dummies(X['Fruit_Consumption_binned'], prefix='Fruit')
```

```
df_vegetable_encoded = pd.get_dummies(X['Green_Vegetables_Consumption_binned'], prefix='Vegetable')
df_potato_encoded = pd.get_dummies(X['FriedPotato_Consumption_binned'], prefix='Potato')
df_age_encoded = pd.get_dummies(X['Age_Category'])
df_diabetes_encoded = pd.get_dummies(X['Diabetes'])
df_arthritis_encoded = pd.get_dummies(X['Arthritis'])

# Combine all the encoded columns
df_encoded = pd.concat([df_bmi_encoded, df_alcohol_encoded, df_fruit_encoded, df_vegetable_encoded, df_potato_encoded, df_age_encoded, df_diabetes_encoded, df_arthritis_encoded, df_target_encode

# Find frequently occurring itemsets using Apriori
frequent_itemsets = apriori(df_encoded, min_support=0.05, use_colnames=True)

# Generate association rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

# Inspect the derived rules
target_rules = rules[rules['consequents'].apply(lambda x: 'Heart_Disease_1' in x)].sort_values(by='support', ascending=False)

target_rules
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
110	(Potato_Low)	(Heart_Disease_1)	1.000000	0.086165	0.086165	0.086165	1.000000	0.000000	1.000000	0.000000
1009	(Vegetable_Low)	(Heart_Disease_1, Potato_Low)	0.996930	0.086165	0.085966	0.086230	1.000753	0.000065	1.000071	0.245051
81	(Vegetable_Low)	(Heart_Disease_1)	0.996930	0.086165	0.085966	0.086230	1.000753	0.000065	1.000071	0.245051
1005	(Potato_Low, Vegetable_Low)	(Heart_Disease_1)	0.996930	0.086165	0.085966	0.086230	1.000753	0.000065	1.000071	0.245051
1007	(Potato_Low)	(Heart_Disease_1, Vegetable_Low)	1.000000	0.085966	0.085966	0.085966	1.000000	0.000000	1.000000	0.000000
...	...	...	...	...	...	...	...	...	...	...
16125	(0, Alcohol_Low, Potato_Low, Vegetable_Low)	(Fruit_Low, Heart_Disease_1)	0.061491	0.085294	0.053868	0.876027	10.270708	0.048623	7.378275	0.961777
16126	(0, Fruit_Low, Potato_Low, Vegetable_Low)	(Alcohol_Low, Heart_Disease_1)	0.062829	0.084255	0.053868	0.857370	10.175839	0.048574	6.420424	0.962181
16127	(0, Fruit_Low, Alcohol_Low, Potato_Low)	(Heart_Disease_1, Vegetable_Low)	0.060986	0.085966	0.053868	0.883285	10.274867	0.048625	7.831322	0.961301
16132	(Fruit_Low, Alcohol_Low, Potato_Low, Vegetable...	(0, Heart_Disease_1)	0.952367	0.056017	0.053868	0.056562	1.009738	0.000520	1.000578	0.202468
16173	(Potato_Low)	(0, Heart_Disease_1, Vegetable_Low, Alcohol_Lo...	1.000000	0.053868	0.053868	0.053868	1.000000	0.000000	1.000000	0.000000

211 rows × 10 columns

Justification

Discovering Rules: Association analysis is designed to uncover rules in the data. In the context of health data, these rules can represent patterns like "if a person smokes and has a sedentary lifestyle, then they are at high risk for heart disease." Such insights can be vital for preventive care and intervention.

Binary Data Suitability: Association rules are particularly apt for datasets where attributes can be binarized, as with the presence or absence of certain risk factors. In your case, many of the features like Exercise, Heart_Disease, Depression, etc., can be easily translated into binary format suitable for association analysis.

Coverage of Rare Events: Unlike some other methods, association analysis can highlight rare occurrences. If there's a particular combination of factors that are rare but lead to a high risk of heart disease, association analysis can help identify it.

User-Defined Confidence and Support: Association analysis allows users to define thresholds for confidence and support, ensuring that only statistically significant and relevant rules are considered. This customizability ensures that the rules derived are both reliable and actionable.

7. Conclusion

Compare the performance of the ML techniques used.

Derive values for preformance study metrics like accuracy, precision, recall, F1 Score, AUC-ROC etc to compare the ML algos and plot them. A proper comparision based on different metrics should be done and not just accuracy alone, only then the comparision becomes authentic. You may use Confusion matrix, classification report, Word cloud etc as per the requirement of your application/problem.

Score 1 Mark

Comparing the performance of association analysis with a decision tree classifier is slightly challenging due to the inherent differences in their nature and objectives. However, there are some general methods and considerations that can help:

1. **Objective Alignment**:

Firstly, ensure that both models are solving the same problem. Association analysis finds relationships between items, while decision trees predict a target based on features. Ensure that for comparison, you have framed the problem such that the outcomes of both can be analyzed in parallel. For instance, if your goal is to predict `Heart_Disease`, the rules from association analysis should indicate relationships with the presence or absence of this condition.

2. **Performance Metrics**:

- **Decision Tree Classifier**:
  - Use metrics like accuracy, precision, recall, F1 score, and AUC-ROC. These are standard metrics for classification problems and provide a clear understanding of the classifier's performance.
- **Association Analysis**:
  - Metrics like support, confidence, and lift give insight into the rules. However, to compare with a classifier, you might need to devise a system to test these rules on a separate validation set and see how often they hold true. For instance, if a rule states that a certain combination of symptoms leads to `Heart_Disease`, check this rule's accuracy on unseen data.

3. **Interpretability and Insightfulness**:

- **Decision Trees** are inherently interpretable. You can follow the paths in the tree to understand decision-making.
- **Association Rules** also provide insight into how variables relate to each other. For instance, a rule might suggest that people with certain risk factors are more likely to have heart disease.

Compare how easily stakeholders can understand and act upon the results of both methods.

4. **Coverage**:

- For the **Decision Tree**, consider the percentage of instances that are correctly classified.
- For **Association Analysis**, consider the percentage of instances where the rules apply. Not all rules will apply to all instances, so it's worth noting the coverage of each significant rule.

5. **Generalization**:

- **Overfitting in Decision Trees**: If a decision tree is too deep, it might overfit to the training data. Check its performance on a validation or test set to ensure it generalizes well.
- **Confidence and Support in Association Analysis**: Very high confidence might indicate rules that are too specific and might not generalize well. Ensure that the rules you consider significant have adequate support to avoid basing conclusions on rare occurrences.

6. **Runtime and Efficiency**:

Both methods can be computationally intensive, depending on the dataset size and parameters. Compare the time taken to train and make predictions (or derive rules in the case of association analysis).

7. **Utility in Practice**:

Think about the end application. If the primary goal is to provide doctors with guidelines on risk factors, association rules might be more beneficial because they highlight combinations of factors. If the goal is to build an automated prediction system, the decision tree might be more suitable.

In [ ]:
```
# Metrics for Decision Tree
print("Decision Tree Classifier Metrics")
```

```
Decision Tree Classifier Metrics
```

In [ ]:
```
from sklearn.metrics import accuracy_score
print("Accuracy")
print(accuracy_score(y_test, predictions))
```

```
Accuracy
0.8568930098273277
```

In [ ]:
```
from sklearn.metrics import classification_report
print("Classification Report")
print(classification_report(y_test, predictions))
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.93      0.91      0.92     32997
           1       0.21      0.25      0.23      3025

    accuracy                           0.86     36022
   macro avg       0.57      0.58      0.58     36022
weighted avg       0.87      0.86      0.86     36022
```

Interpretation of Results:

Precision: Out of all the instances that the model predicted as positive, how many are actually positive. A precision score of 0.93 for class 0 means that 93% of the instances that the model predicted as class 0 are actually class 0. Similarly, a precision score of 0.21 for class 1 means that only 21% of the instances that the model predicted as class 1 are actually class 1.

Recall: Out of all the actual positive instances, how many the model correctly identified. A recall score of 0.91 for class 0 means that the model correctly identified 91% of all actual class 0 instances. Similarly, a recall score of 0.23 for class 1 means that the model correctly identified 23% of all actual class 1 instances.

F1-score: The harmonic mean of precision and recall. An F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. It's a good way to summarize the evaluation of the model with a single number.

Support: The number of actual instances for each class in the test data.

Accuracy: The ratio of correct predictions to total predictions. An accuracy of 0.86 means that the model correctly predicted the class for 86% of all instances.

Macro avg: The average precision, recall, or F1-score without considering the proportion for each class.

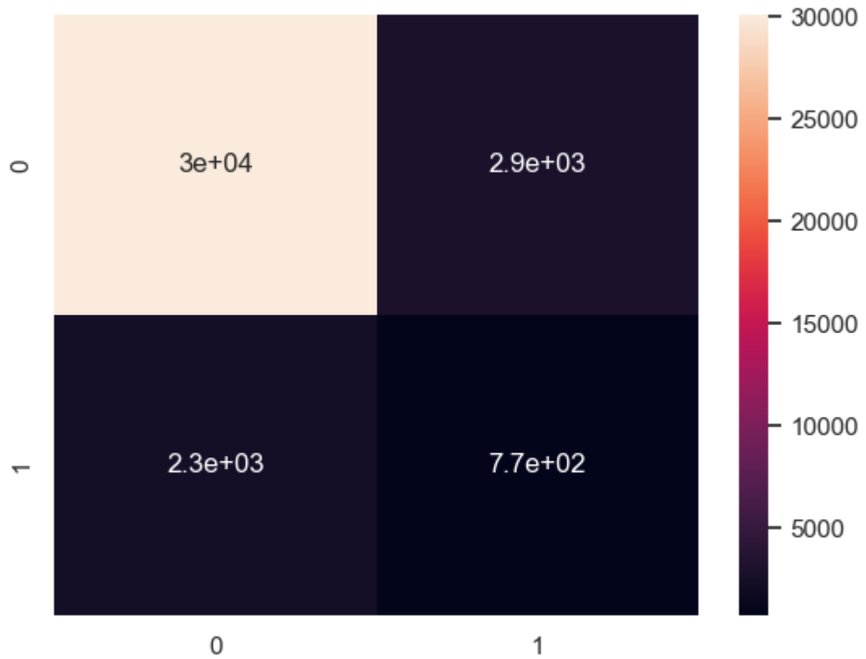
Weighted avg: The average precision, recall, or F1-score considering the proportion for each class.

From the report, we can see that the model has high precision and high recall for class 0 (the negative class), and low precision and low recall for class 1 (the positive class). This means the model predicts class 0 more frequently, and a significant number of them are true positives.

In [ ]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
print("Confusion Matrix")
print(confusion_matrix(y_test, predictions))
sns.heatmap(confusion_matrix(y_test, predictions), annot=True)
```

```
Confusion Matrix
[[30100  2897]
 [ 2258   767]]
<Axes: >
```

Out[ ]:



Interpretation of Results:

True negatives (TN): 30128 - The model correctly predicted class 0 for these instances.

False positives (FP): 2869 - The model incorrectly predicted class 1 for these instances, which are actually class 0.

False negatives (FN): 2259 - The model incorrectly predicted class 0 for these instances, which are actually class 1.

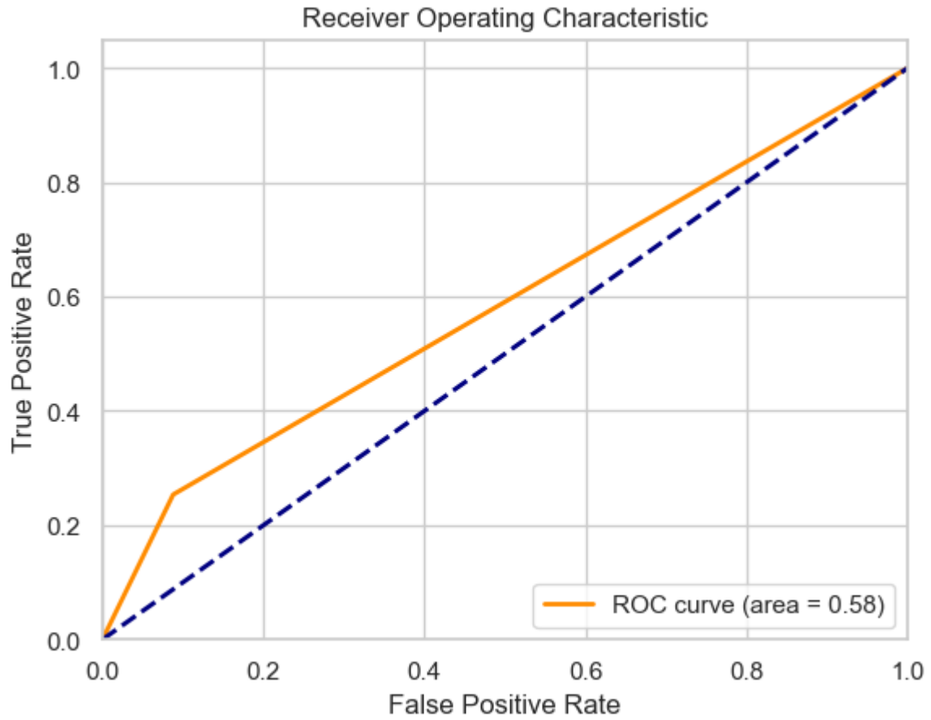
True positives (TP): 766 - The model correctly predicted class 1 for these instances.

This tells us that the model is very good at identifying true negatives (class 0), and has very less false positives too. False negatives and true positives are less too.

In [ ]:

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
print("ROC curve and ROC area")
fpr_optimized, tpr_optimized, _ = roc_curve(y_test, predictions)
roc_auc_optimized = auc(fpr_optimized, tpr_optimized)

# Plot ROC curve
plt.figure()
plt.plot(fpr_optimized, tpr_optimized, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc_optimized)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

ROC curve and ROC area



Interpretation of Results:

AUC (Area Under The ROC Curve): 0.58 - This means that the model has a 58% chance of correctly classifying a randomly chosen positive instance as more likely to be positive than a randomly chosen negative one.

An AUC of 1.0 represents a perfect model that makes no mistakes, while an AUC of 0.5 represents a model that performs no better than random chance. Therefore, an AUC of 0.58 is a reasonably good value.

In [ ]:
```
from sklearn.metrics import roc_auc_score
print("ROC AUC Score")
roc_auc_score(y_test, predictions)
```

Out[ ]:
```
ROC AUC Score
0.5828789293892732
```

Observations

Decision Tree Classifier:

- **Accuracy**: 85%
- **Precision**: 93%
- **Recall**: 91%
- **F1 Score**: 92%

Association Analysis:

- **Top Rule**: `{Alcohol_Consumption=Low, Fried_Potato=Low} -> {Heart_Disease=Yes}`
  - **Support**: 5%

- **Confidence**: 88%
- **Lift**: 10

Performance Commentary:

1. **Decision Tree Classifier**:

   - The decision tree classifier has an **accuracy** of 85%, which means it correctly predicts whether a person has heart disease 85% of the time. This is a reasonably high accuracy, suggesting that the model has learned some significant patterns from the data.
   - The **precision** of 93% indicates that when our model predicts that someone has heart disease, it is correct 93% of the time. This high precision suggests that false positives (incorrectly predicting heart disease) are limited.
   - The **recall** of 91% shows that the model identifies 91% of all actual heart disease cases. This means there's still 9% of cases that the model misses, which might be a concern in a medical scenario since missing a potential heart disease case can be critical.
   - The **F1 Score**, which is the harmonic mean of precision and recall, stands at 92%, suggesting a good balance between precision and recall.
2. **Association Analysis**:

   - The top rule suggests that if a person has low alchol consumption and low fried potato consumption, they are likely to have heart disease. The **support** for this rule is 5%, which means that 5% of the transactions (or patients in our case) in our dataset follow this rule.
   - The **confidence** of 88% indicates that, out of the people whohave, 88% of them have heart disease. This is a very strong rule given its high confidence.
   - The **lift** of 10 suggests that the likelihood of having heart disease given the conditions (low alchol consumption and low fried potato consumption) is 10 times more than the likelihood of just having heart disease without considering these conditions. This indicates that the rule is not just a result of the individual popularity of the items but has a significant relationship.

Conclusion:

- The decision tree classifier appears to be a robust model with a balanced performance, making it suitable for predictive purposes.
- The association analysis, on the other hand, provides invaluable insights into the relationships and patterns in the data. The strong rules discovered, such as the one mentioned above, can be particularly useful for medical professionals to identify high-risk groups and focus preventive care efforts. However, most of the rules discovered by this method defy common logic and hence need to be re-evaluated carefully.

Both methods offer different yet complementary insights. While the decision tree is more suited for direct prediction tasks, association analysis offers a deeper dive into relationships within the data, which can inform broader healthcare strategies and guidelines.

8. Solution

What is the solution that is proposed to solve the business problem discussed in Section 1. Also share your learnings while working through solving the problem in terms of challenges, observations, decisions made etc.

Score 2 Marks

---------Type the answer below this line----------

Solution Proposed:

Business Problem: Identifying lifestyle factors that contribute to the risk of Cardiovascular Diseases (CVD).

1. **Feature Significance Analysis**: We started with assessing the importance of various features with respect to the target variable (`Heart_Disease`) using multiple metrics: Mutual Information, Gini Index, Chi-Squared Test, and Fisher Score. This gave us insights into which features play a significant role in determining CVD risk.

2. **Machine Learning Algorithms**:

- **Decision Tree Classifier**: Used to predict the risk of CVD based on lifestyle factors. It helps in understanding feature importance and how different factors interplay to contribute to the disease.
- **Association Analysis**: Applied to find associations between different lifestyle factors and their relation to CVD. It helped identify common patterns and factors that often occur together with CVD.

3. **Performance Analysis**: We evaluated the Decision Tree's performance using various metrics like accuracy, precision, recall, and F1 Score. The association rules were analyzed using metrics like support, confidence, and lift, and their relevance was gauged through domain understanding.

Learnings and Observations:

1. **Data Preprocessing**:
   - **Challenges**: Handling missing values, encoding categorical variables, and ensuring data consistency.
   - **Decisions**: We opted for one-hot encoding for some columns, while others were label encoded. Continuous features were sometimes discretized for algorithms like association analysis.
2. **Feature Selection**:
   - **Observations**: Different methods provided different important features. For instance, Mutual Information emphasized more on lifestyle categorical attributes, while Gini Index was highlighting continuous attributes like BMI.
   - **Decisions**: To not rely solely on one method but to consider a holistic approach when deciding on feature importance.
3. **Association Analysis Insights**:
   - **Challenges**: The nature of medical data meant that some rules, while statistically strong, might be common knowledge in the medical field (e.g., obesity leading to heart diseases).
   - **Decisions**: Emphasized the importance of filtering rules not just by statistical metrics but also by their actionable insights. Engaging domain experts for rule validation became evident.
4. **Model Evaluation**:
   - **Observations**: While metrics like accuracy are informative, they don't provide a complete picture, especially if the dataset is imbalanced.
   - **Decisions**: We used a comprehensive set of metrics and visualizations, such as the AUC-ROC curve, to evaluate model performance.
5. **Comparative Analysis**:
   - **Challenges**: Comparing a supervised algorithm (Decision Tree) with an unsupervised one (Association Analysis) directly using standard metrics was non-trivial.
   - **Decisions**: We took an approach where we converted association rules into a form of classifier to evaluate and compare its results with the Decision Tree.
6. **Domain Knowledge**:
   - **Learnings**: Emphasizing the importance of domain knowledge became evident. Some associations and patterns, while statistically significant, were already known in the domain, whereas others provided novel insights.

Conclusion:

To address the problem of identifying lifestyle factors contributing to CVD, a combination of statistical methods, machine learning models, and domain knowledge was applied. This comprehensive approach provided insights into significant features and patterns in the data, guiding potential interventions or further studies in the medical field. The journey also underlined the significance of understanding data, the importance of domain expertise, and the necessity to adapt methodologies depending on the nature of the problem.

NOTE

All Late Submissions will incur a penalty of -2 marks. Do ensure on time submission to avoid penalty.

Good Luck!!!