ALLIANCE UNIVERSITY

Private University established in Karnataka State by Act No.34 of year 2010
Recognized by the University Grants Commission (UGC), New Delhi

PROJECT REPORT

Bachelor of Computer Applications

SEMESTER – II

STATISTICS FOR DATA SCIENCE

CARDIOVASCULAR DISEASE ANALYSIS using STATISTICAL techniques

Submitted By :

Anuska Ghosh

BCA – A

Registration Number: 2411021240012

Department of Computer Applications

Alliance University

Chandapura – Anekal Main Road, Anekal

Bengaluru – 562 106

April 2025

NAME: ANUSKA GHOSH

REGISTRATION NUMBER: 2411021240012

SUBJECT: STATISTICS FOR DATA SCIENCE

GITHUB REPO LINK: https://github.com/anuskaghosh17/SDS_Project

PROJECT TITLE: Cardiovascular Disease Analysis Using Statistical Techniques

Table of Contents

1. Introduction

Cardiovascular disease (CVD) remains one of the leading causes of morbidity and mortality worldwide. With increasing lifestyle-related risk factors such as poor diet, lack of exercise, smoking, and alcohol consumption, early identification of individuals at risk has become crucial in public health and medical practice.

In this project, we analyze a comprehensive dataset that captures demographic, lifestyle, and health information of individuals. The goal is to apply statistical techniques to uncover patterns, test hypotheses, and develop predictive models that can aid in better understanding the factors associated with CVD.

Through the application of the CRISP-DM framework and statistical tools, we aim to transform raw data into meaningful insights and actionable knowledge.

2. Project Goals

The main objective of this project is to apply statistical techniques and data science methodologies to analyze a health dataset and derive meaningful insights related to cardiovascular disease (CVD). Specifically, this project aims to:

- **Understand the structure and characteristics** of the dataset, including variables related to general health, exercise habits, chronic conditions, and dietary patterns.
- **Perform descriptive and inferential statistical analysis** to identify significant relationships between variables and the occurrence of heart disease.
- **Explore the distribution** of key lifestyle and medical attributes (e.g., BMI, alcohol consumption, smoking history) and assess their association with CVD.
- **Apply probability concepts and hypothesis testing** to validate assumptions and answer research questions statistically.
- **Engineer meaningful features** and select the most relevant ones for predictive modeling.
- **Build and evaluate statistical models** to predict the likelihood of having heart disease based on the available features.
- **Present visual insights** using data visualizations to support data-driven storytelling.
- **Reflect on real-world implications** of the findings and suggest how such analysis can assist in preventive healthcare strategies.

3. Dataset Overview

The dataset used in this project is titled CVD_cleaned.csv and contains health, demographic, and lifestyle information for 308,854 individuals. The goal of this dataset is to help identify factors associated with cardiovascular disease and other chronic conditions.

Dataset Dimensions:

- **Rows (Individuals):** 308,854
- **Columns (Features):** 19
- **Data Types:** 12 categorical (object), 7 numerical (float)

Demographic Features:

- **Age_Category:** Age groups (18-24, 25-29, ..., 80+)
- **Sex:** Male/Female
- **Height_(cm):** Height in centimeters
- **Weight_(kg):** Weight in kilograms
- **BMI:** Body Mass Index (calculated from height/weight)

Column Name: Description

- **General_Health:** Self-assessed overall health status (e.g., Excellent, Good, Poor)
- **Checkup:** Frequency of medical checkups
- **Exercise:** Whether the person exercises or not
- **Heart_Disease:** Indicates if the person has heart disease (Yes or No)
- **Skin_Cancer:** Whether diagnosed with skin cancer
- **Other_Cancer:** Whether diagnosed with any cancer other than skin cancer
- **Depression:** Whether diagnosed with depression
- **Diabetes:** Indicates if the person has diabetes
- **Arthritis:** Whether diagnosed with arthritis
- **Sex:** Biological sex (Male or Female)
- **Age_Category:** Age group (e.g., 18-24, 25-29, ..., 80+)
- **Height_(cm):** Height in centimeters
- **Weight_(kg):** Weight in kilograms
- **BMI:** Body Mass Index (calculated from height and weight)
- **Smoking_History:** Indicates if the person has a history of smoking
- **Alcohol_Consumption:** Amount of alcohol consumed (unit to be clarified; possibly per month)
- **Fruit_Consumption:** Frequency or servings of fruit consumed (likely per month)
- **Green_Vegetables_Consumption:** Frequency of green vegetable consumption
- **FriedPotato_Consumption:** Frequency of consuming fried potatoes (e.g., fries, chips)

This dataset provides a comprehensive view of various physical, behavioral, and medical attributes of individuals. The target variable for this project is Heart_Disease, which is binary (Yes/No) and indicates whether the individual has a cardiovascular condition.

4. CRISP-DM Framework:

The CRISP-DM (Cross-Industry Standard Process for Data Mining) framework provides a structured approach to data science projects. This project follows each step of CRISP-DM to ensure thorough analysis and insight generation.

1. **BUSINESS UNDERSTANDING**

- **Objective:** Understand which factors are associated with heart disease and use them to build predictive models.

- **Key Question:** What are the most significant health and lifestyle indicators that increase the risk of cardiovascular disease?

1. **DATA UNDERSTANDING**

- The dataset was explored to understand the types of variables present, data quality, and preliminary patterns.

- Summary statistics and data visualizations helped form initial hypotheses and identify necessary preprocessing steps.

1. **DATA PREPARATION**

- **Data was cleaned and formatted:** no missing values were found.

- Categorical variables were encoded and numerical variables were scaled or normalized where appropriate.

1. **MODELING**

- Various statistical models such as logistic regression and decision trees were applied.
- The target variable for prediction is Heart_Disease.

1. **EVALUATION**

- Models were evaluated based on accuracy, precision, recall, and F1-score.
- Cross-validation was used to assess model stability and generalizability.

1. **DEPLOYMENT**

- Although this project is academic, the results and models could be deployed in a healthcare setting to flag high-risk individuals for early intervention.

⚠ 5. Challenges Faced

During the development of this project, several challenges arose that required thoughtful solutions:

1. **Lack of Unit Clarification**

- Variables like Alcohol_Consumption, Fruit_Consumption, and FriedPotato_Consumption lacked clearly defined units (e.g., per week or per month).
- **Resolution:** Assumed they represent monthly consumption based on value ranges, but this uncertainty was noted for interpretation and modeling.

1. **Imbalanced Target Variable**

- The Heart_Disease variable showed class imbalance (more “No” than “Yes”), which could bias model predictions.
- **Resolution:** Used stratified sampling and balanced evaluation metrics like precision, recall, and F1-score to assess model performance more fairly.

1. **High Correlation Between Features**

- Features like Height_(cm), Weight_(kg), and BMI are mathematically related, possibly leading to multicollinearity.
- **Resolution:** Evaluated correlation matrix and considered dropping one of the correlated features or using dimensionality reduction if necessary.

1. **Categorical Encoding**

- Many categorical variables needed transformation into numerical format for modeling.
- **Resolution:** Applied label encoding and one-hot encoding as appropriate during data preprocessing.

1. **Generalization vs Overfitting**

- Some complex models initially performed well on training data but poorly on test data.
- **Resolution:** Implemented cross-validation and regularization to reduce overfitting.

6. Data Preprocessing

STEP1: IMPORTING THE REQUIRED LIBRARIES

Why?

These are standard Python libraries for:

a> Data manipulation (pandas, numpy)

b> Data visualization (matplotlib, seaborn)

c> Data preprocessing, modeling, and evaluation (sklearn)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, MinMaxScaler,
StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, precision_score, recall_score
```

STEP2: LOAD THE DATASET

Why?

This loads the dataset into a pandas DataFrame

```
df = pd.read_csv(r"C:\Users\anusk\Downloads\CVD_cleaned.csv (1)\
CVD_cleaned.csv")
df
```

```
       General_Health                  Checkup Exercise Heart_Disease
\
0                Poor  Within the past 2 years       No            No

1           Very Good     Within the past year       No           Yes

2           Very Good     Within the past year      Yes            No

3                Poor     Within the past year      Yes           Yes
```

```
4                Good     Within the past year       No             No

...               ...                       ...      ...            ...

308849      Very Good     Within the past year      Yes             No

308850           Fair  Within the past 5 years      Yes             No

308851      Very Good      5 or more years ago      Yes             No

308852      Very Good     Within the past year      Yes             No

308853      Excellent     Within the past year      Yes             No


       Skin_Cancer Other_Cancer Depression  \
0               No           No         No
1               No           No         No
2               No           No         No
3               No           No         No
4               No           No         No
...            ...          ...        ...
308849          No           No         No
308850          No           No         No
308851          No           No        Yes
308852          No           No         No
308853          No           No         No

                                          Diabetes Arthritis
Sex  \
0                                           No       Yes  Female

1                                          Yes        No  Female

2                                          Yes        No  Female

3                                          Yes        No    Male

4                                           No        No    Male

...                                        ...       ...     ...

308849                                      No        No    Male

308850                                     Yes        No    Male

308851  Yes, but female told only during pregnancy        No  Female

308852                                      No        No    Male

308853                                      No        No  Female
```

```
       Age_Category  Height_(cm)  Weight_(kg)    BMI
Smoking_History  \
0             70-74        150.0        32.66  14.54              Yes

1             70-74        165.0        77.11  28.29               No

2             60-64        163.0        88.45  33.47               No

3             75-79        180.0        93.44  28.73               No

4               80+        191.0        88.45  24.37              Yes

...              ...          ...          ...    ...              ...

308849        25-29        168.0        81.65  29.05               No

308850        65-69        180.0        69.85  21.48               No

308851        30-34        157.0        61.23  24.69              Yes

308852        65-69        183.0        79.38  23.73               No

308853        45-49        160.0        81.19  31.71               No


        Alcohol_Consumption  Fruit_Consumption
Green_Vegetables_Consumption  \
0                       0.0               30.0
16.0
1                       0.0               30.0
0.0
2                       4.0               12.0
3.0
3                       0.0               30.0
30.0
4                       0.0                8.0
4.0
...                     ...                ...
...
308849                  4.0               30.0
8.0
308850                  8.0               15.0
60.0
308851                  4.0               40.0
8.0
308852                  3.0               30.0
12.0
308853                  1.0                5.0
12.0
```

```
        FriedPotato_Consumption
0                          12.0
1                           4.0
2                          16.0
3                           8.0
4                           0.0
...                         ...
308849                      0.0
308850                      4.0
308851                      4.0
308852                      0.0
308853                      1.0

[308854 rows x 19 columns]
```

Displaying the first 5 rows

```
df.head()
```

```
  General_Health                   Checkup Exercise Heart_Disease
Skin_Cancer  \
0           Poor  Within the past 2 years       No             No
No
1      Very Good     Within the past year       No            Yes
No
2      Very Good     Within the past year      Yes             No
No
3           Poor     Within the past year      Yes            Yes
No
4           Good     Within the past year       No             No
No

  Other_Cancer Depression Diabetes Arthritis     Sex Age_Category  \
0           No         No       No       Yes  Female        70-74
1           No         No      Yes        No  Female        70-74
2           No         No      Yes        No  Female        60-64
3           No         No      Yes        No    Male        75-79
4           No         No       No        No    Male          80+

   Height_(cm)  Weight_(kg)    BMI Smoking_History
Alcohol_Consumption  \
0        150.0        32.66  14.54             Yes
0.0
1        165.0        77.11  28.29              No
0.0
2        163.0        88.45  33.47              No
4.0
3        180.0        93.44  28.73              No
0.0
```

```
4        191.0        88.45  24.37              Yes
0.0

   Fruit_Consumption  Green_Vegetables_Consumption
FriedPotato_Consumption
0               30.0                          16.0
12.0
1               30.0                           0.0
4.0
2               12.0                           3.0
16.0
3               30.0                          30.0
8.0
4                8.0                           4.0
0.0
```

Displaying the last 5 rows

```
df.tail()

       General_Health                  Checkup Exercise Heart_Disease
\
308849      Very Good     Within the past year      Yes             No

308850           Fair  Within the past 5 years      Yes             No

308851      Very Good      5 or more years ago      Yes             No

308852      Very Good     Within the past year      Yes             No

308853      Excellent     Within the past year      Yes             No


       Skin_Cancer Other_Cancer Depression  \
308849          No           No         No
308850          No           No         No
308851          No           No        Yes
308852          No           No         No
308853          No           No         No

                                          Diabetes Arthritis
Sex  \
308849                                        No        No    Male

308850                                       Yes        No    Male

308851  Yes, but female told only during pregnancy        No  Female

308852                                        No        No    Male
```

```
308853                                          No         No  Female


       Age_Category  Height_(cm)  Weight_(kg)    BMI
Smoking_History  \
308849        25-29        168.0        81.65  29.05              No

308850        65-69        180.0        69.85  21.48              No

308851        30-34        157.0        61.23  24.69             Yes

308852        65-69        183.0        79.38  23.73              No

308853        45-49        160.0        81.19  31.71              No


        Alcohol_Consumption  Fruit_Consumption
Green_Vegetables_Consumption  \
308849                  4.0               30.0
8.0
308850                  8.0               15.0
60.0
308851                  4.0               40.0
8.0
308852                  3.0               30.0
12.0
308853                  1.0                5.0
12.0

        FriedPotato_Consumption
308849                      0.0
308850                      4.0
308851                      4.0
308852                      0.0
308853                      1.0
```

STEP 3: SUMMARY AND STATISTICAL SUMMARY OF THE DATASET

df.info(): Gives info about the dataset like data types, non-null counts, and memory usage. Helps identify missing values and data types.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 308854 entries, 0 to 308853
Data columns (total 19 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   General_Health                308854 non-null  object
 1   Checkup                       308854 non-null  object
```

```
 2   Exercise                      308854 non-null  object
 3   Heart_Disease                 308854 non-null  object
 4   Skin_Cancer                   308854 non-null  object
 5   Other_Cancer                  308854 non-null  object
 6   Depression                    308854 non-null  object
 7   Diabetes                      308854 non-null  object
 8   Arthritis                     308854 non-null  object
 9   Sex                           308854 non-null  object
 10  Age_Category                  308854 non-null  object
 11  Height_(cm)                   308854 non-null  float64
 12  Weight_(kg)                   308854 non-null  float64
 13  BMI                           308854 non-null  float64
 14  Smoking_History               308854 non-null  object
 15  Alcohol_Consumption           308854 non-null  float64
 16  Fruit_Consumption             308854 non-null  float64
 17  Green_Vegetables_Consumption  308854 non-null  float64
 18  FriedPotato_Consumption       308854 non-null  float64
dtypes: float64(7), object(12)
memory usage: 44.8+ MB
```

df.describe(): Provides summary statistics for both numerical and categorical columns (mean, std, count, top, freq, etc.)

```
# For Numerical Columns
df.describe()
```

```
         Height_(cm)    Weight_(kg)             BMI
Alcohol_Consumption  \
count  308854.000000  308854.000000  308854.000000
308854.000000
mean      170.615249      83.588655      28.626211
5.096366
std        10.658026      21.343210       6.522323
8.199763
min        91.000000      24.950000      12.020000
0.000000
25%       163.000000      68.040000      24.210000
0.000000
50%       170.000000      81.650000      27.440000
1.000000
75%       178.000000      95.250000      31.850000
6.000000
max       241.000000     293.020000      99.330000
30.000000

       Fruit_Consumption  Green_Vegetables_Consumption  \
count      308854.000000                 308854.000000
mean           29.835200                     15.110441
std            24.875735                     14.926238
```

```
min              0.000000                      0.000000
25%             12.000000                      4.000000
50%             30.000000                     12.000000
75%             30.000000                     20.000000
max            120.000000                    128.000000
```

```
       FriedPotato_Consumption
count             308854.000000
mean                   6.296616
std                    8.582954
min                    0.000000
25%                    2.000000
50%                    4.000000
75%                    8.000000
max                  128.000000
```

```
# For both Numerical and Categorical Columns
df.describe(include='all')
```

```
       General_Health                Checkup Exercise Heart_Disease  \
count          308854                308854   308854        308854
unique              5                     5        2             2
top         Very Good  Within the past year      Yes            No
freq           110395                239371   239381        283883
mean              NaN                   NaN      NaN           NaN
std               NaN                   NaN      NaN           NaN
min               NaN                   NaN      NaN           NaN
25%               NaN                   NaN      NaN           NaN
50%               NaN                   NaN      NaN           NaN
75%               NaN                   NaN      NaN           NaN
max               NaN                   NaN      NaN           NaN

       Skin_Cancer Other_Cancer Depression Diabetes Arthritis     Sex
\
count       308854       308854     308854   308854    308854  308854

unique           2            2          2        4         2       2

top             No           No         No       No        No  Female

freq        278860       278976     246953   259141    207783  160196

mean            NaN          NaN        NaN      NaN       NaN     NaN

std             NaN          NaN        NaN      NaN       NaN     NaN

min             NaN          NaN        NaN      NaN       NaN     NaN

25%             NaN          NaN        NaN      NaN       NaN     NaN

50%             NaN          NaN        NaN      NaN       NaN     NaN
```

```
75%              NaN           NaN         NaN       NaN         NaN      NaN

max              NaN           NaN         NaN       NaN         NaN      NaN


       Age_Category    Height_(cm)    Weight_(kg)            BMI  \
count        308854  308854.000000  308854.000000  308854.000000
unique           13            NaN            NaN            NaN
top           65-69            NaN            NaN            NaN
freq          33434            NaN            NaN            NaN
mean            NaN     170.615249      83.588655      28.626211
std             NaN      10.658026      21.343210       6.522323
min             NaN      91.000000      24.950000      12.020000
25%             NaN     163.000000      68.040000      24.210000
50%             NaN     170.000000      81.650000      27.440000
75%             NaN     178.000000      95.250000      31.850000
max             NaN     241.000000     293.020000      99.330000

       Smoking_History  Alcohol_Consumption  Fruit_Consumption  \
count           308854        308854.000000      308854.000000
unique               2                   NaN                NaN
top                 No                   NaN                NaN
freq            183590                   NaN                NaN
mean               NaN              5.096366          29.835200
std                NaN              8.199763          24.875735
min                NaN              0.000000           0.000000
25%                NaN              0.000000          12.000000
50%                NaN              1.000000          30.000000
75%                NaN              6.000000          30.000000
max                NaN             30.000000         120.000000

       Green_Vegetables_Consumption  FriedPotato_Consumption
count                  308854.000000            308854.000000
unique                           NaN                      NaN
top                              NaN                      NaN
freq                             NaN                      NaN
mean                       15.110441                 6.296616
std                        14.926238                 8.582954
min                         0.000000                 0.000000
25%                         4.000000                 2.000000
50%                        12.000000                 4.000000
75%                        20.000000                 8.000000
max                       128.000000               128.000000
```

STEP 4: CHECK FOR MISSING VALUES

df.isnull().sum(): This checks how many null values exist in each column. We'll handle them if found.

```
df.isnull().sum()
```

```
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
```

Output Interpretation:

After checking for missing values using df.isnull().sum(), we found that there are no missing values in the dataset. This indicates that the dataset is clean in this aspect, and no imputation techniques are required for further processing.

STEP 5: ENCODE CATEGORICAL VARIABLES

- We map Yes/No fields to 1/0 for machine learning compatibility.
- We one-hot encode multi-category columns like General_Health, Checkup, and Age_Category to convert them into dummy variables.

```
# Binary columns to encode as 0/1
binary_cols = ['Exercise', 'Heart_Disease', 'Skin_Cancer',
'Other_Cancer',
               'Depression', 'Diabetes', 'Arthritis',
'Smoking_History']

# Encode binary columns
df[binary_cols] = df[binary_cols].apply(lambda x: x.map({'Yes': 1,
'No': 0}))

# Encode 'Sex' as Male = 1, Female = 0
df['Sex'] = df['Sex'].map({'Male': 1, 'Female': 0})

# One-hot encode multi-category columns
```

```
df = pd.get_dummies(df, columns=['General_Health', 'Checkup',
'Age_Category'], drop_first=True)

df
```

```
        Exercise  Heart_Disease  Skin_Cancer  Other_Cancer  Depression
\
0              0              0            0             0           0

1              0              1            0             0           0

2              1              0            0             0           0

3              1              1            0             0           0

4              0              0            0             0           0

...          ...            ...          ...           ...         ...

308849         1              0            0             0           0

308850         1              0            0             0           0

308851         1              0            0             0           1

308852         1              0            0             0           0

308853         1              0            0             0           0


        Diabetes  Arthritis  Sex  Height_(cm)  Weight_(kg)  ...  \
0            0.0          1    0        150.0        32.66  ...
1            1.0          0    0        165.0        77.11  ...
2            1.0          0    0        163.0        88.45  ...
3            1.0          0    1        180.0        93.44  ...
4            0.0          0    1        191.0        88.45  ...
...          ...        ...  ...          ...          ...  ...
308849       0.0          0    1        168.0        81.65  ...
308850       1.0          0    1        180.0        69.85  ...
308851       NaN          0    0        157.0        61.23  ...
308852       0.0          0    1        183.0        79.38  ...
308853       0.0          0    0        160.0        81.19  ...

        Age_Category_35-39  Age_Category_40-44  Age_Category_45-49  \
0                    False               False               False
1                    False               False               False
2                    False               False               False
3                    False               False               False
4                    False               False               False
...                    ...                 ...                 ...
308849               False               False               False
```

```
308850                False                False                False
308851                False                False                False
308852                False                False                False
308853                False                False                 True

        Age_Category_50-54  Age_Category_55-59  Age_Category_60-64  \
0                    False                False                False
1                    False                False                False
2                    False                False                 True
3                    False                False                False
4                    False                False                False
...                    ...                  ...                  ...
308849               False                False                False
308850               False                False                False
308851               False                False                False
308852               False                False                False
308853               False                False                False

        Age_Category_65-69  Age_Category_70-74  Age_Category_75-79  \
0                    False                True                False
1                    False                True                False
2                    False                False               False
3                    False                False                True
4                    False                False               False
...                    ...                  ...                  ...
308849               False                False               False
308850                True                False               False
308851               False                False               False
308852                True                False               False
308853               False                False               False

        Age_Category_80+
0                  False
1                  False
2                  False
3                  False
4                   True
...                  ...
308849             False
308850             False
308851             False
308852             False
308853             False

[308854 rows x 36 columns]
```

STEP 6: FEATURE SCALING

- We standardize continuous variables to have a mean of 0 and standard deviation of 1.

- This is especially useful for algorithms like logistic regression or KNN that are sensitive to scale.

```
# Columns to scale
scale_cols = ['Height_(cm)', 'Weight_(kg)', 'BMI',
              'Alcohol_Consumption', 'Fruit_Consumption',
              'Green_Vegetables_Consumption',
'FriedPotato_Consumption']

scaler = StandardScaler()
df[scale_cols] = scaler.fit_transform(df[scale_cols])

df[scale_cols]
```

```
        Height_(cm)  Weight_(kg)       BMI  Alcohol_Consumption  \
0         -1.934250    -2.386180 -2.159696            -0.621527
1         -0.526857    -0.303547 -0.051548            -0.621527
2         -0.714510     0.227770  0.742649            -0.133707
3          0.880535     0.461569  0.015913            -0.621527
4          1.912623     0.227770 -0.652562            -0.621527
...             ...          ...       ...                  ...
308849    -0.245379    -0.090833  0.064975            -0.133707
308850     0.880535    -0.643702 -1.095656             0.354113
308851    -1.277466    -1.047579 -0.603499            -0.133707
308852     1.162014    -0.197190 -0.750686            -0.255662
308853    -0.995988    -0.112385  0.472806            -0.499572

        Fruit_Consumption  Green_Vegetables_Consumption  \
0                0.006625                      0.059597
1                0.006625                     -1.012342
2               -0.716973                     -0.811354
3                0.006625                      0.997544
4               -0.877772                     -0.744358
...                   ...                           ...
308849           0.006625                     -0.476373
308850          -0.596373                      3.007431
308851           0.408624                     -0.476373
308852           0.006625                     -0.208388
308853          -0.998372                     -0.208388

        FriedPotato_Consumption
0                      0.664502
1                     -0.267579
2                      1.130543
3                      0.198462
4                     -0.733620
...                         ...
308849                -0.733620
308850                -0.267579
308851                -0.267579
```

```
308852                -0.733620
308853                -0.617110

[308854 rows x 7 columns]

df[scale_cols].describe()

        Height_(cm)   Weight_(kg)           BMI
Alcohol_Consumption  \
count  3.088540e+05  3.088540e+05  3.088540e+05         3.088540e+05

mean  -5.860032e-16  2.120213e-16 -1.192666e-15        -6.257572e-17

std    1.000002e+00  1.000002e+00  1.000002e+00         1.000002e+00

min   -7.469993e+00 -2.747419e+00 -2.546062e+00        -6.215270e-01

25%   -7.145095e-01 -7.285071e-01 -6.770928e-01        -6.215270e-01

50%   -5.772641e-02 -9.083253e-02 -1.818696e-01        -4.995721e-01

75%    6.928828e-01  5.463735e-01  4.942709e-01         1.102026e-01

max    6.603931e+00  9.812567e+00  1.084029e+01         3.037121e+00


       Fruit_Consumption  Green_Vegetables_Consumption  \
count       3.088540e+05                  3.088540e+05
mean        5.742243e-17                 -5.769850e-17
std         1.000002e+00                  1.000002e+00
min        -1.199372e+00                 -1.012342e+00
25%        -7.169729e-01                 -7.443576e-01
50%         6.624922e-03                 -2.083878e-01
75%         6.624922e-03                  3.275820e-01
max         3.624614e+00                  7.563174e+00

       FriedPotato_Consumption
count             3.088540e+05
mean              5.374150e-17
std               1.000002e+00
min              -7.336199e-01
25%              -5.005995e-01
50%              -2.675792e-01
75%               1.984616e-01
max               1.417968e+01
```

STEP 7: FINAL DATASET OVERVIEW AFTER PREPROCESSING

After preprocessing, we check the final shape and preview the cleaned dataset, which is now fully numerical and ready for EDA and modeling.

```
print("Shape of dataset:", df.shape)
```

```
Shape of dataset: (308854, 36)
```

7. Exploratory Data Analysis (EDA)

Goal of EDA:

To understand the distribution of variables, identify patterns, detect outliers, and explore relationships between features — especially with the target variable Heart_Disease.

SUMMARY STATISTICS

We use .describe() to understand central tendencies (mean, median), spread (std, min, max), and identify any potential outliers.

```
# Summary statistics of numerical features
df.describe()
```

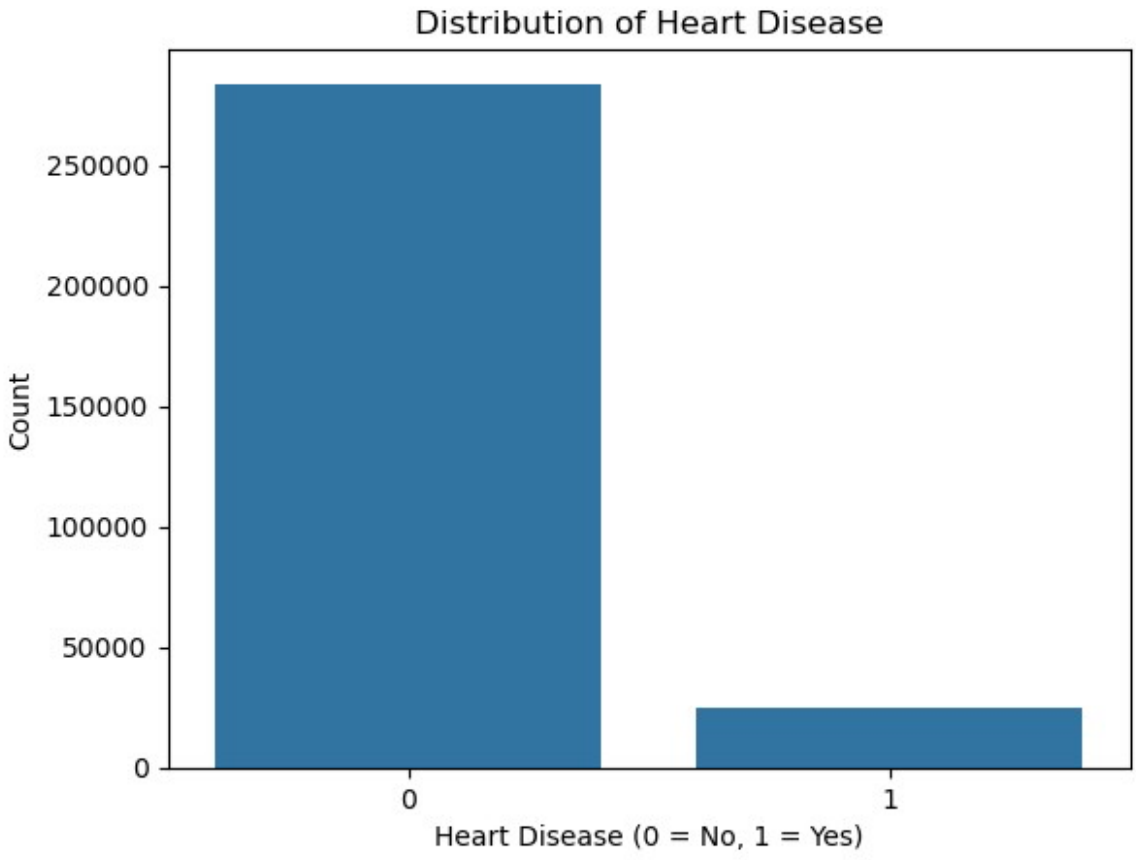
```
            Exercise  Heart_Disease    Skin_Cancer   Other_Cancer  \
count  308854.000000  308854.000000  308854.000000  308854.000000   
mean        0.775062       0.080850       0.097114       0.096738   
std         0.417542       0.272606       0.296113       0.295602   
min         0.000000       0.000000       0.000000       0.000000   
25%         1.000000       0.000000       0.000000       0.000000   
50%         1.000000       0.000000       0.000000       0.000000   
75%         1.000000       0.000000       0.000000       0.000000   
max         1.000000       1.000000       1.000000       1.000000   

          Depression       Diabetes      Arthritis            Sex  \
count  308854.000000  299312.000000  308854.000000  308854.000000   
mean        0.200422       0.134211       0.327245       0.481321   
std         0.400316       0.340880       0.469208       0.499652   
min         0.000000       0.000000       0.000000       0.000000   
25%         0.000000       0.000000       0.000000       0.000000   
50%         0.000000       0.000000       0.000000       0.000000   
75%         0.000000       0.000000       1.000000       1.000000   
max         1.000000       1.000000       1.000000       1.000000   

        Height_(cm)   Weight_(kg)            BMI  Smoking_History  \
count  3.088540e+05  3.088540e+05  3.088540e+05    308854.000000   
mean  -5.860032e-16  2.120213e-16 -1.192666e-15         0.405577   
std    1.000002e+00  1.000002e+00  1.000002e+00         0.491004   
min   -7.469993e+00 -2.747419e+00 -2.546062e+00         0.000000   
25%   -7.145095e-01 -7.285071e-01 -6.770928e-01         0.000000   
50%   -5.772641e-02 -9.083253e-02 -1.818696e-01         0.000000   
75%    6.928828e-01  5.463735e-01  4.942709e-01         1.000000   
max    6.603931e+00  9.812567e+00  1.084029e+01         1.000000   

       Alcohol_Consumption  Fruit_Consumption
```

```
Green_Vegetables_Consumption  \
count          3.088540e+05       3.088540e+05
3.088540e+05
mean          -6.257572e-17       5.742243e-17                  -
5.769850e-17
std            1.000002e+00       1.000002e+00
1.000002e+00
min           -6.215270e-01      -1.199372e+00                  -
1.012342e+00
25%           -6.215270e-01      -7.169729e-01                  -
7.443576e-01
50%           -4.995721e-01       6.624922e-03                  -
2.083878e-01
75%            1.102026e-01       6.624922e-03
3.275820e-01
max            3.037121e+00       3.624614e+00
7.563174e+00

       FriedPotato_Consumption
count             3.088540e+05
mean              5.374150e-17
std               1.000002e+00
min              -7.336199e-01
25%              -5.005995e-01
50%              -2.675792e-01
75%               1.984616e-01
max               1.417968e+01
```

DISTRIBUTION OF THE TARGET VARIABLE

We visualize the count of people with and without heart disease. We also calculate the percentage of each class to check for imbalance.

```
# Distribution of Heart Disease
sns.countplot(data=df, x='Heart_Disease')
plt.title("Distribution of Heart Disease")
plt.xlabel("Heart Disease (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.show()

# Calculate class balance
heart_disease_ratio = df['Heart_Disease'].value_counts(normalize=True)
heart_disease_ratio
```

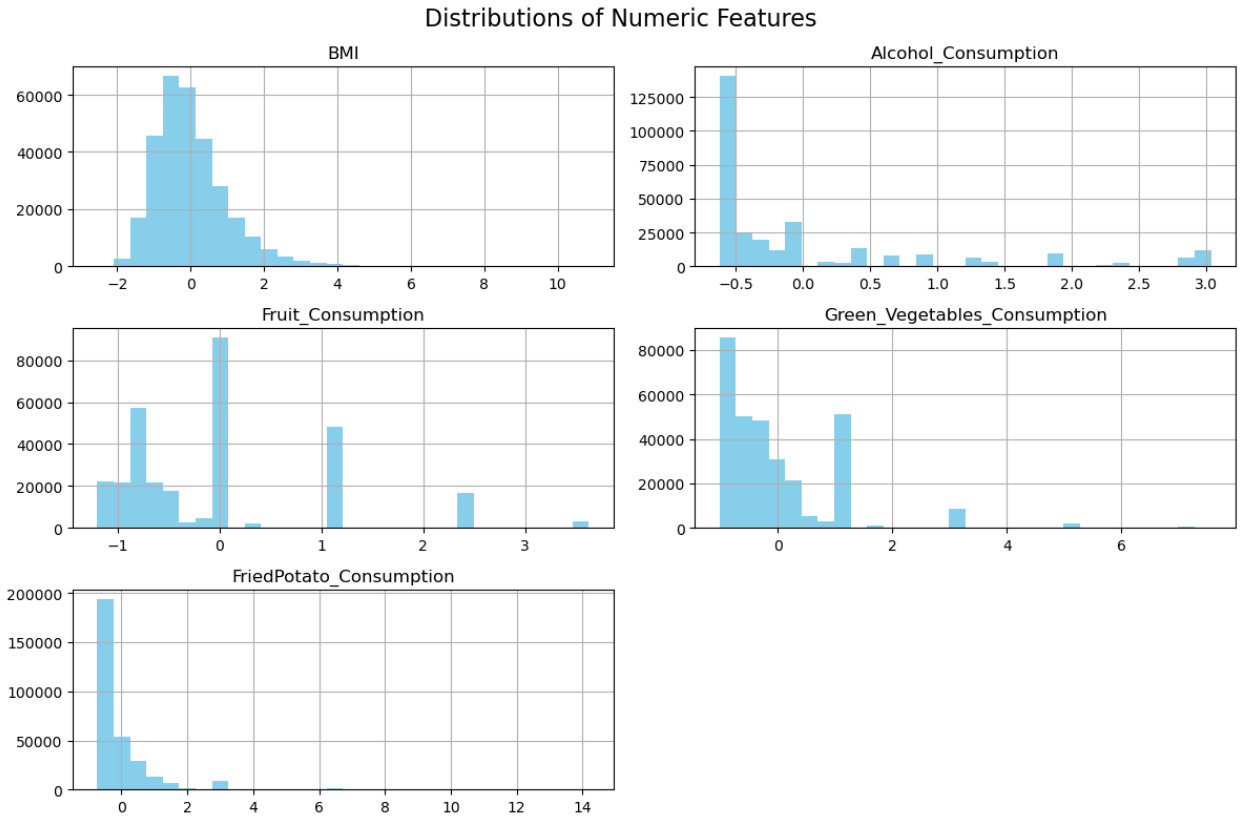



```
Heart_Disease
0    0.91915
1    0.08085
Name: proportion, dtype: float64
```

UNIVARIATE ANALYSIS – NUMERIC VARIABLES

This gives us a sense of how each numeric variable is distributed — skewed, normal, or bimodal.

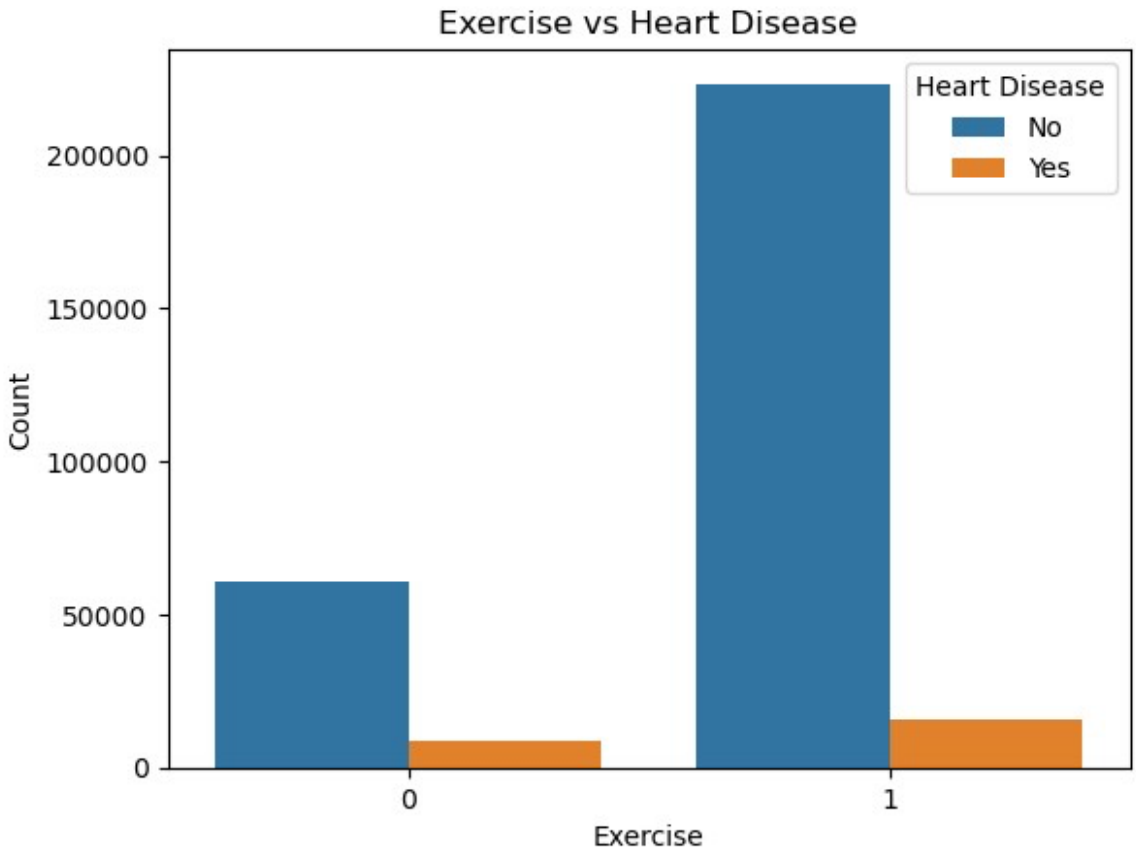
```
# Plot histograms for selected numeric features
num_cols = ['BMI', 'Alcohol_Consumption', 'Fruit_Consumption',
            'Green_Vegetables_Consumption', 'FriedPotato_Consumption']

df[num_cols].hist(figsize=(12, 8), bins=30, color='skyblue')
plt.suptitle("Distributions of Numeric Features", fontsize=16)
plt.tight_layout()
plt.show()
```

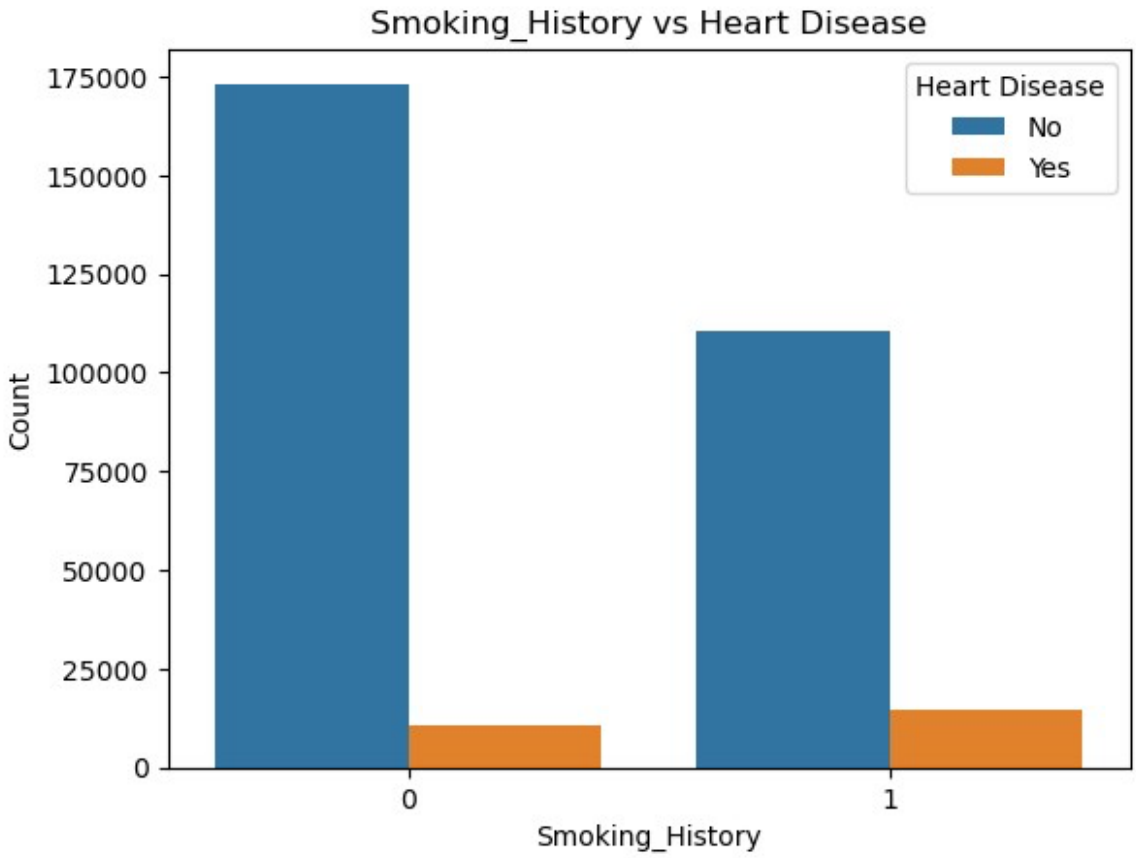


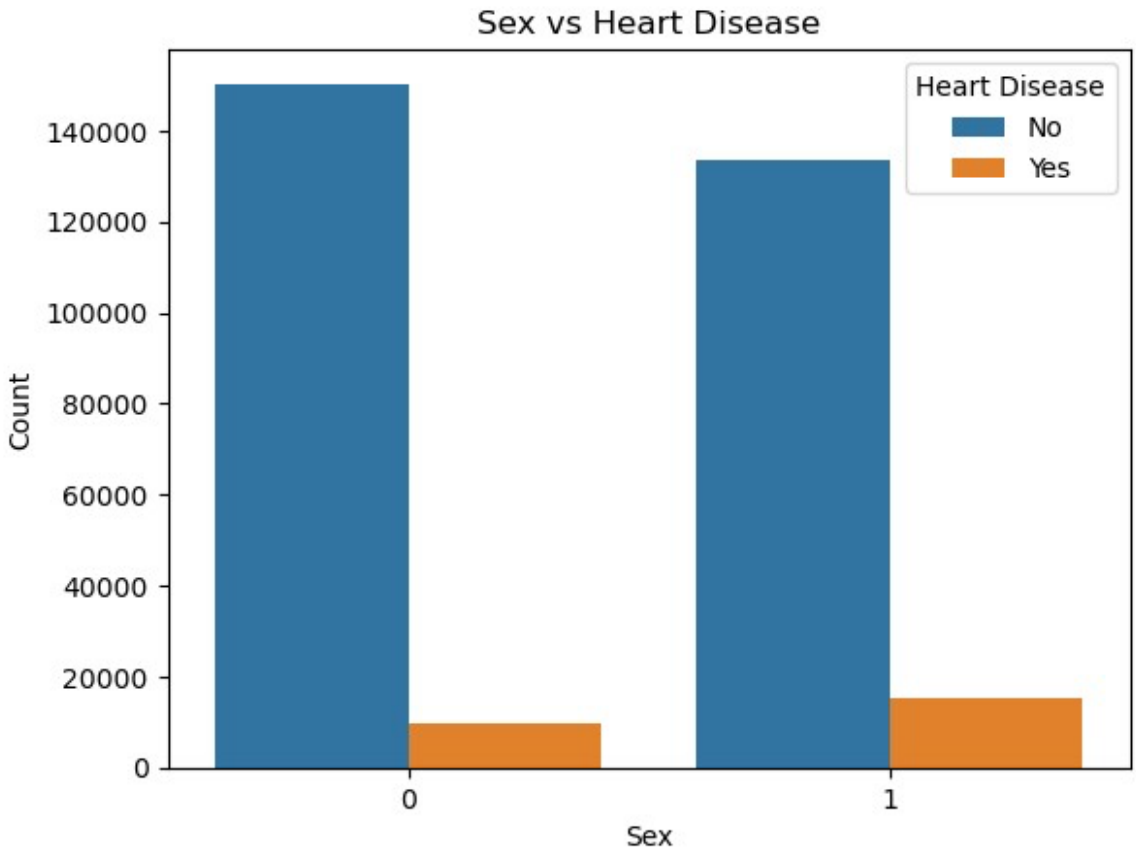
BIVARIATE ANALYSIS – CATEGORICAL vs TARGET

This shows how heart disease prevalence varies across different categories. For example, do smokers or sedentary individuals have higher heart disease rates?

```
# Relationship between categorical features and heart disease
categorical = ['Exercise', 'Smoking_History', 'Sex']

for col in categorical:
    sns.countplot(data=df, x=col, hue='Heart_Disease')
    plt.title(f"{col} vs Heart Disease")
    plt.ylabel("Count")
    plt.legend(title='Heart Disease', labels=['No', 'Yes'])
    plt.show()
```


Exercise vs Heart Disease
Heart Disease
No
Yes
200000
150000
100000
50000
0
Count
0
1
Exercise


Smoking_History vs Heart Disease
Heart Disease
No
Yes
175000
150000
125000
100000
75000
50000
25000
0
Count
0
1
Smoking_History

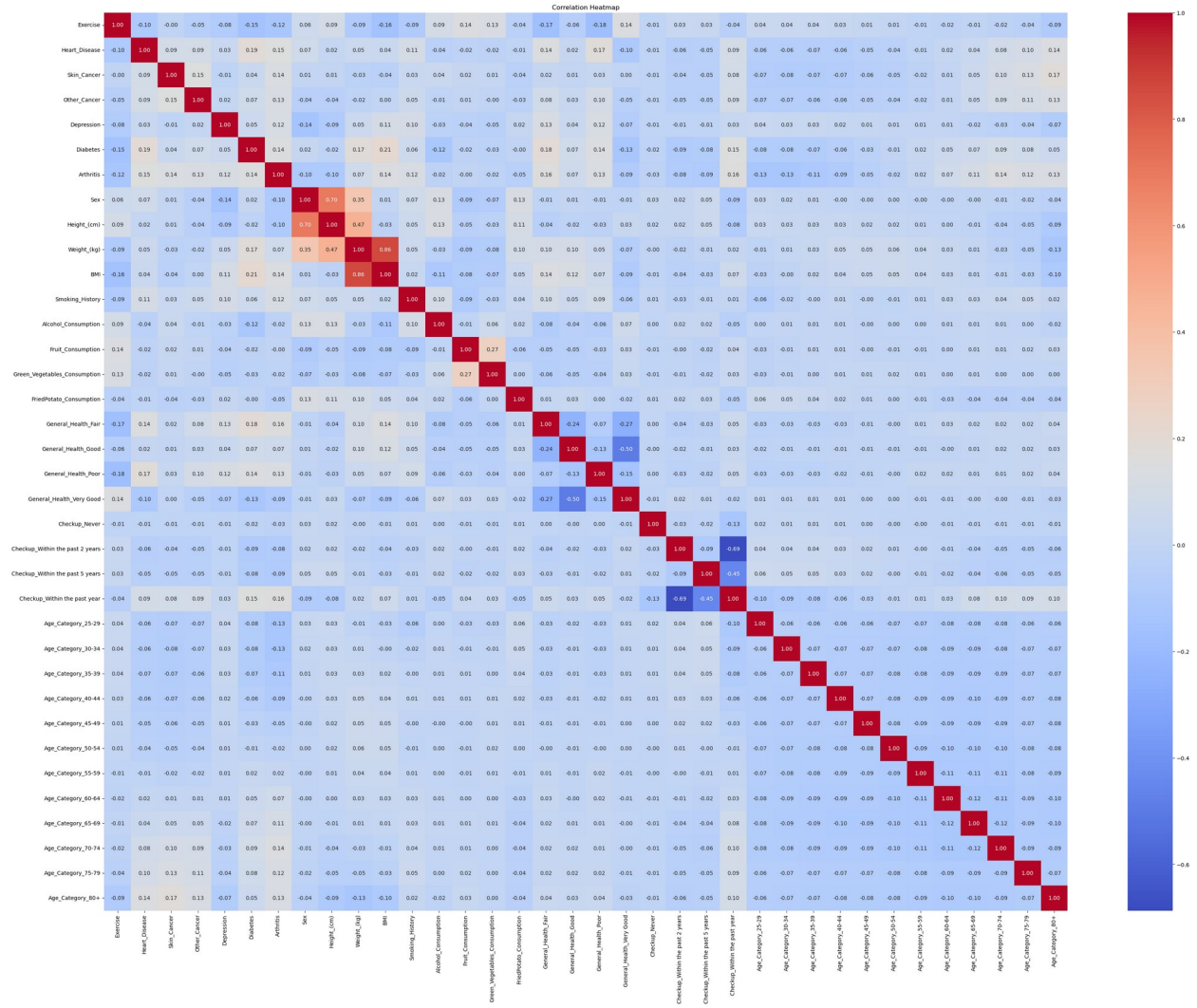


CORRELATION MATRIX

We generate a heatmap to identify correlations between variables. This helps to:

1. Spot multicollinearity (e.g., BMI vs Weight/Height)
2. Discover relationships between features and the target

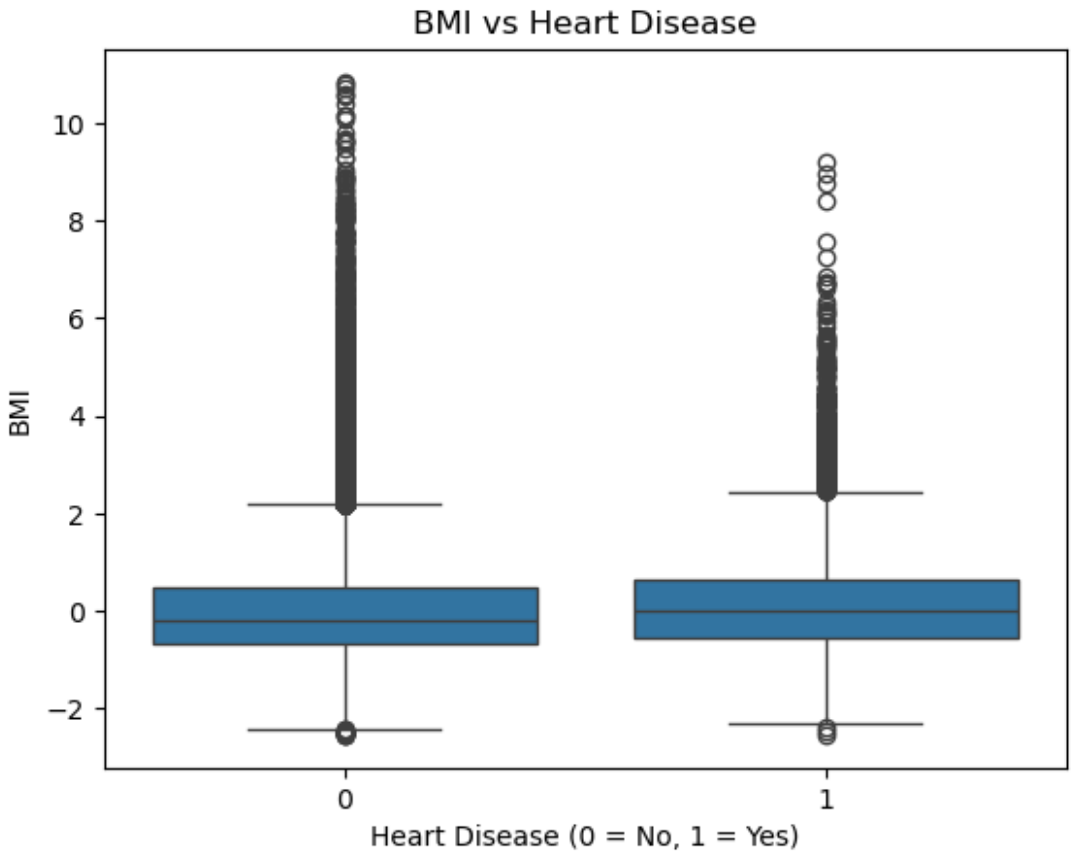
```
# Correlation matrix for numeric features
plt.figure(figsize=(40, 30))
corr_matrix = df.corr()
sns.heatmap(corr_matrix, cmap="coolwarm", annot=True, fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



BOXPLOTS FOR OUTLIERS & PATTERNS

Boxplots reveal outliers and whether higher/lower BMI is associated with heart disease.

```
# Boxplot for BMI by Heart Disease
sns.boxplot(data=df, x='Heart_Disease', y='BMI')
plt.title("BMI vs Heart Disease")
plt.xlabel("Heart Disease (0 = No, 1 = Yes)")
plt.ylabel("BMI")
plt.show()
```



OUTLIER REMOVAL USING IQR METHOD(INTERQUARTILE RANGE)

- Q1 (25th percentile) and Q3 (75th percentile) are used to define the Interquartile Range (IQR).
- Any data point below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR is considered an outlier.
- We loop through the specified numeric columns and filter out rows with outliers.

This method preserves the integrity of your dataset while removing extreme values that could skew statistical analysis or model performance.

```
# Function to remove outliers using IQR method
def remove_outliers_iqr(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Keep only the rows within the IQR bounds
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
```

```
    return df

# Columns to check for outliers (numerical features)
numeric_cols = ['BMI', 'Alcohol_Consumption', 'Fruit_Consumption',
                'Green_Vegetables_Consumption',
'FriedPotato_Consumption']

# Remove outliers
df_cleaned = remove_outliers_iqr(df, numeric_cols)
df_cleaned
```

	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression
0	0	0	0	0	0
1	0	1	0	0	0
2	1	0	0	0	0
3	1	1	0	0	0
4	0	0	0	0	0
...	...	...	...	...	...
308848	1	0	0	0	0
308849	1	0	0	0	0
308851	1	0	0	0	1
308852	1	0	0	0	0
308853	1	0	0	0	0

	Diabetes	Arthritis	Sex	Height_(cm)	Weight_(kg)	...	\
0	0.0	1	0	-1.934250	-2.386180	...	
1	1.0	0	0	-0.526857	-0.303547	...	
2	1.0	0	0	-0.714510	0.227770	...	
3	1.0	0	1	0.880535	0.461569	...	
4	0.0	0	1	1.912623	0.227770	...	
...	...	...	...	...	...	...	
308848	0.0	0	1	-0.245379	-1.153467	...	
308849	0.0	0	1	-0.245379	-0.090833	...	
308851	NaN	0	0	-1.277466	-1.047579	...	
308852	0.0	0	1	1.162014	-0.197190	...	
308853	0.0	0	0	-0.995988	-0.112385	...	

	Age_Category_35-39	Age_Category_40-44	Age_Category_45-49	\
0	False	False	False	

```
1                    False                False                False
2                    False                False                False
3                    False                False                False
4                    False                False                False
...                    ...                  ...                  ...
308848               False                False                False
308849               False                False                False
308851               False                False                False
308852               False                False                False
308853               False                False                 True

        Age_Category_50-54  Age_Category_55-59  Age_Category_60-64  \
0                    False                False                False
1                    False                False                False
2                    False                False                 True
3                    False                False                False
4                    False                False                False
...                    ...                  ...                  ...
308848               False                 True                False
308849               False                False                False
308851               False                False                False
308852               False                False                False
308853               False                False                False

        Age_Category_65-69  Age_Category_70-74  Age_Category_75-79  \
0                    False                 True                False
1                    False                 True                False
2                    False                False                False
3                    False                False                 True
4                    False                False                False
...                    ...                  ...                  ...
308848               False                False                False
308849               False                False                False
308851               False                False                False
308852                True                False                False
308853               False                False                False

        Age_Category_80+
0                  False
1                  False
2                  False
3                  False
4                   True
...                  ...
308848             False
308849             False
308851             False
308852             False
308853             False
```

```
[189271 rows x 36 columns]

# Check new shape
print("Shape before outlier removal:", df.shape)
print("Shape after outlier removal:", df_cleaned.shape)

Shape before outlier removal: (308854, 36)
Shape after outlier removal: (189271, 36)
```

8. Univariate, Bivariate & Multivariate Analysis

This section helps us statistically understand individual variables (univariate), their relationships with the target (bivariate), and interactions between multiple variables (multivariate).

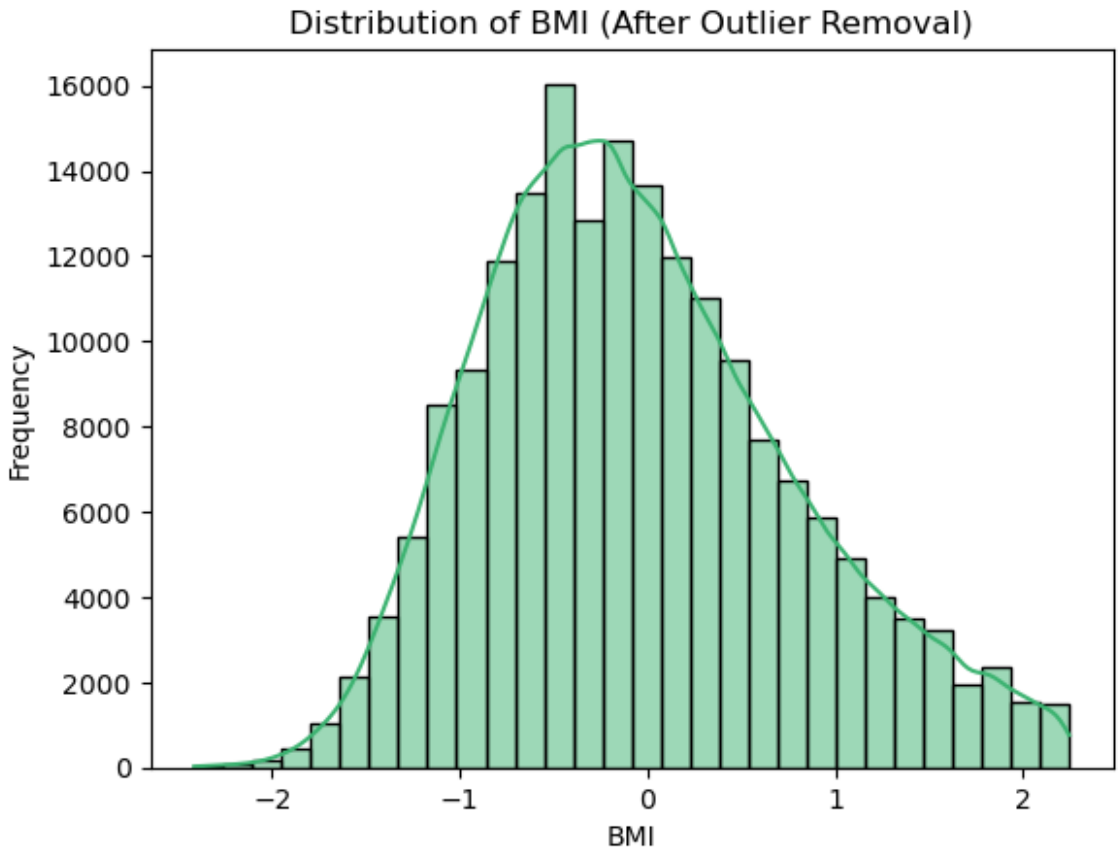
We'll use the outlier-removed dataset: df_cleaned.

UNIVARIATE ANALYSIS: Study of a single variable at a time

1. Distribution of BMI

This helps us understand the shape of BMI distribution — is it normal, skewed, or uniform?

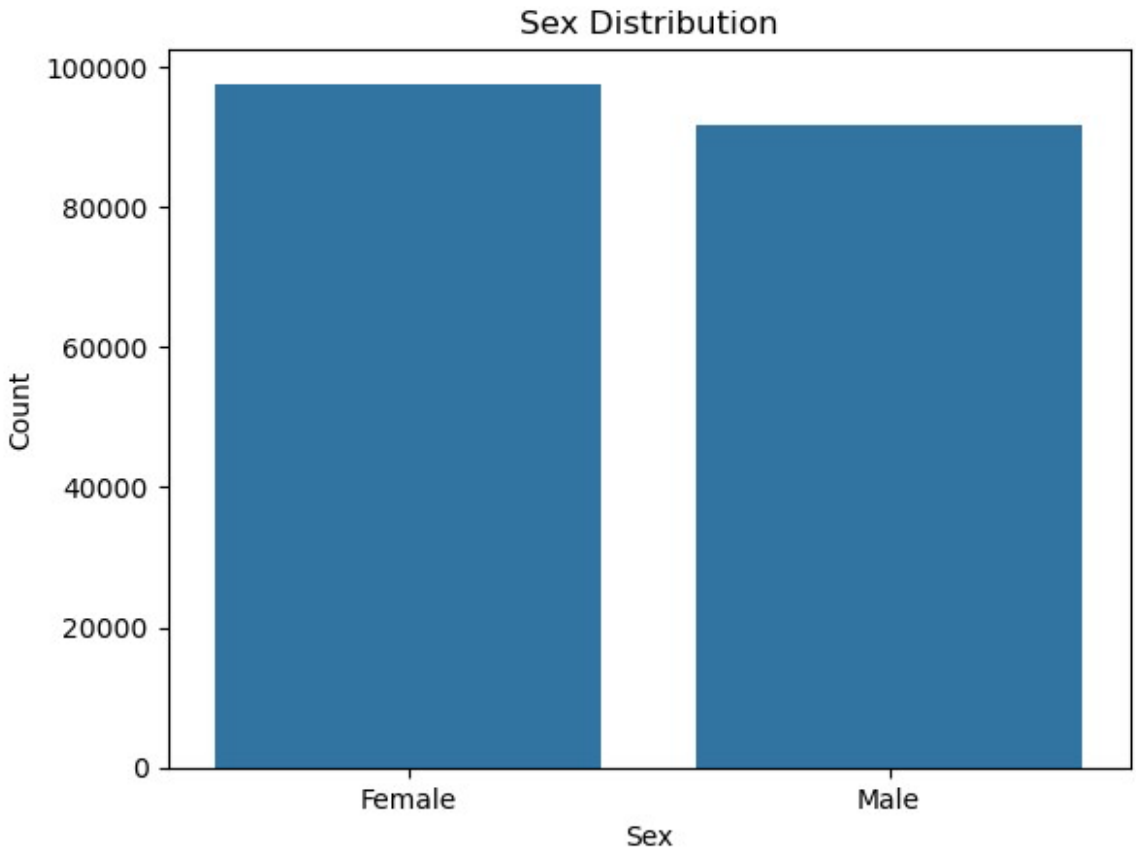
```
sns.histplot(df_cleaned['BMI'], kde=True, bins=30,
color='mediumseagreen')
plt.title("Distribution of BMI (After Outlier Removal)")
plt.xlabel("BMI")
plt.ylabel("Frequency")
plt.show()
```



2. Count of People by Sex

Gives a quick view of gender proportions in the dataset.

```
sns.countplot(data=df_cleaned, x='Sex')
plt.title("Sex Distribution")
plt.xticks([0, 1], labels=["Female", "Male"])
plt.ylabel("Count")
plt.show()
```

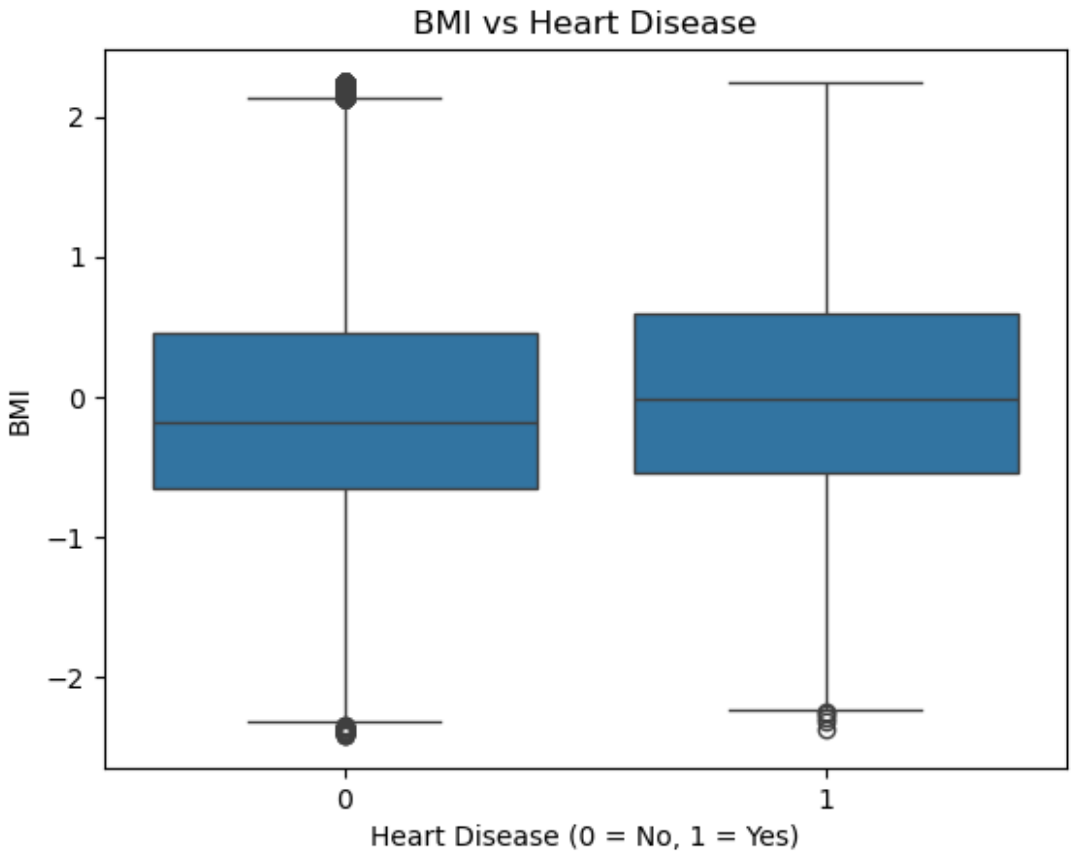


BIVARIATE ANALYSIS: One independent variable vs. one dependent variable

1. BMI vs Heart Disease

Helps spot whether people with heart disease tend to have higher BMI.

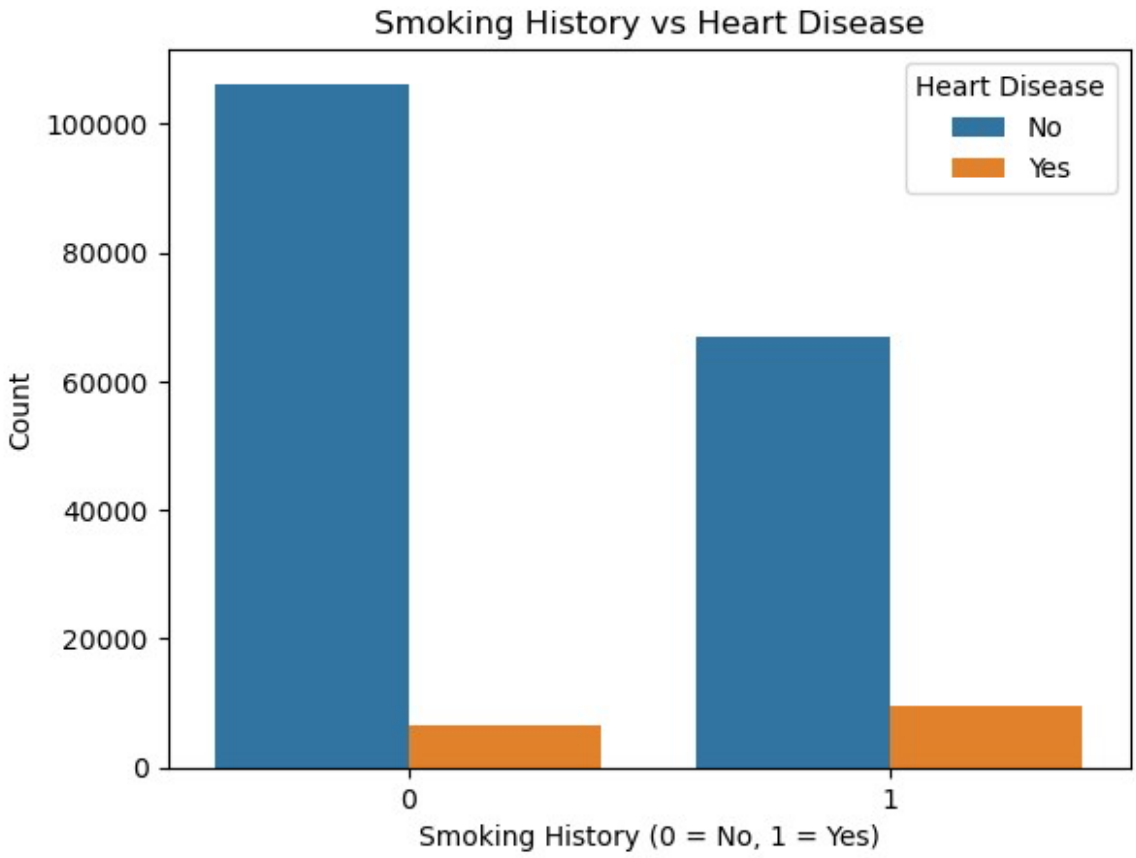
```
sns.boxplot(data=df_cleaned, x='Heart_Disease', y='BMI')
plt.title("BMI vs Heart Disease")
plt.xlabel("Heart Disease (0 = No, 1 = Yes)")
plt.ylabel("BMI")
plt.show()
```



2. Smoking vs Heart Disease

This explores the link between smoking and cardiovascular risk.

```
sns.countplot(data=df_cleaned, x='Smoking_History',
hue='Heart_Disease')
plt.title("Smoking History vs Heart Disease")
plt.xlabel("Smoking History (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.legend(title='Heart Disease', labels=['No', 'Yes'])
plt.show()
```

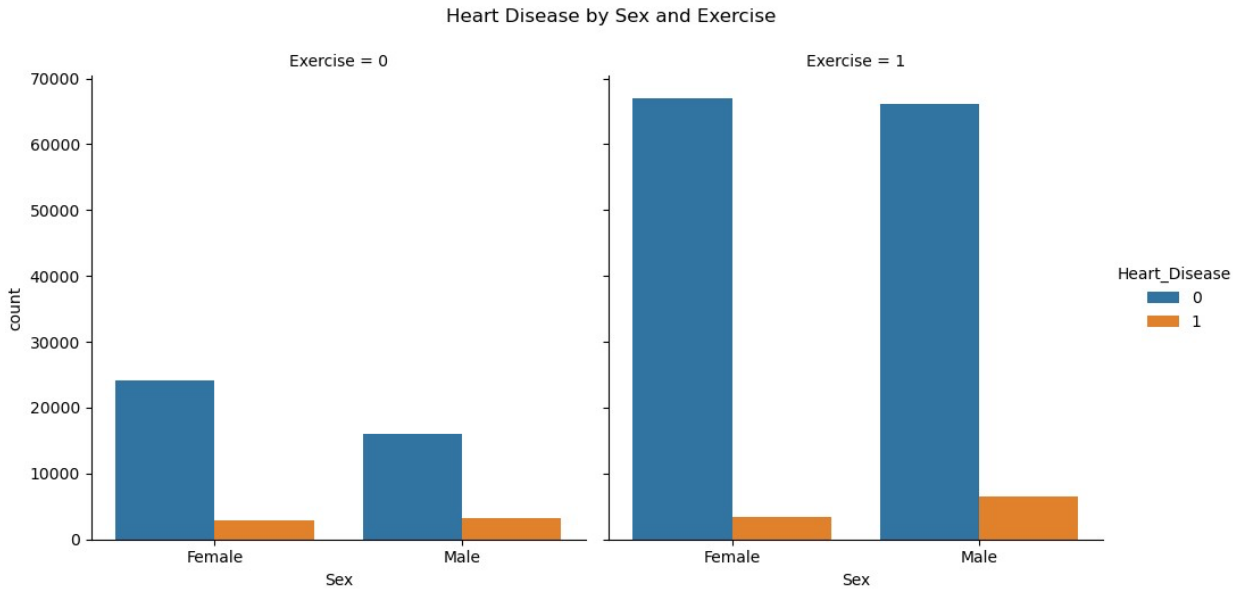


MULTIVARIATE ANALYSIS: Looking at 3 or more variables together

1. Heart Disease by Sex and Exercise

This shows how gender and exercise frequency interact in relation to heart disease presence.

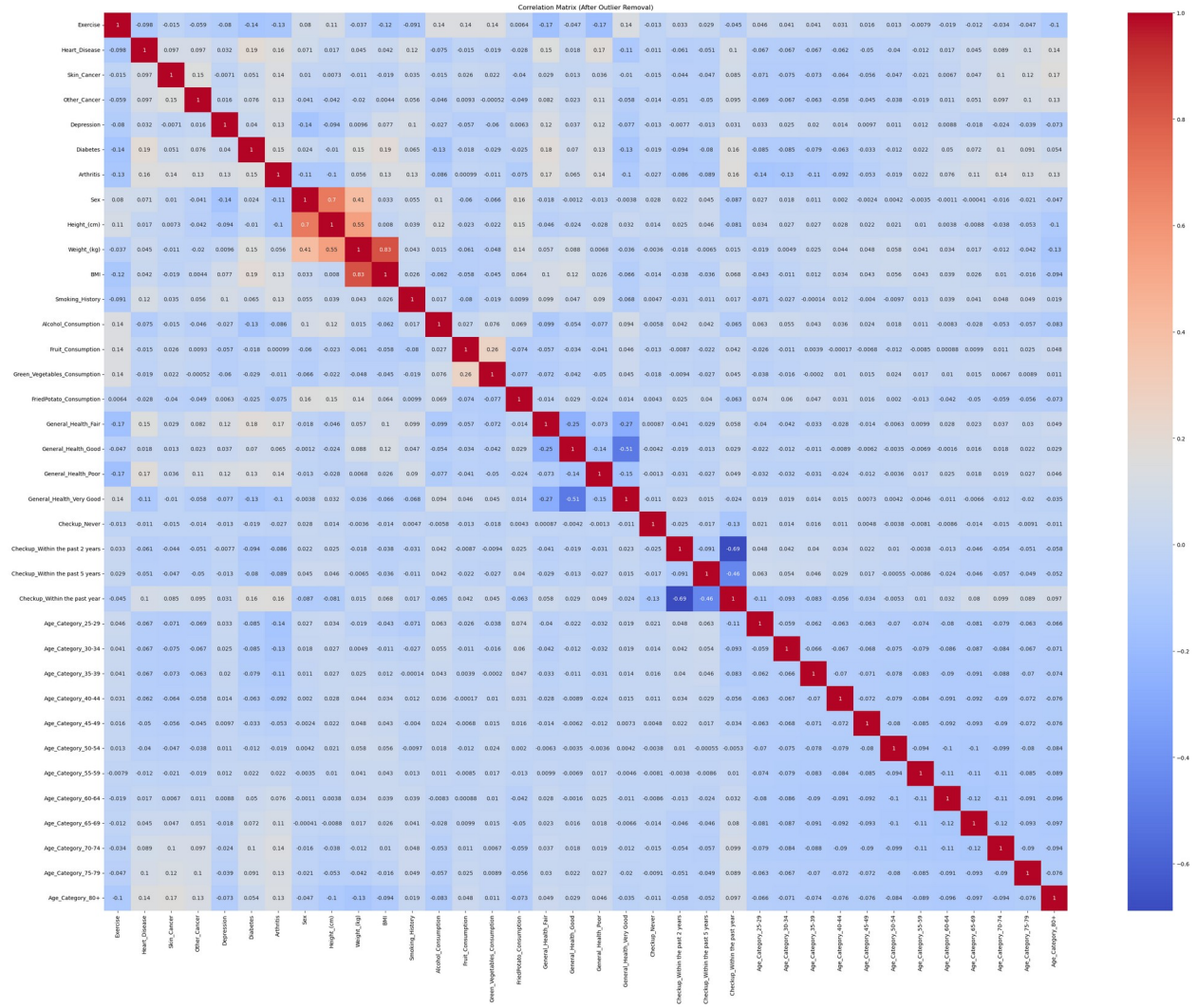
```
sns.catplot(data=df_cleaned, x='Sex', hue='Heart_Disease',
col='Exercise', kind='count')
plt.suptitle("Heart Disease by Sex and Exercise", y=1.05)
plt.xticks([0, 1], ["Female", "Male"])
plt.show()
```



2. Correlation Heatmap (Cleaned Data)

We analyze how strongly features are related to each other — especially looking for predictors highly correlated with Heart_Disease.

```
plt.figure(figsize=(40, 30))
sns.heatmap(df_cleaned.corr(), cmap='coolwarm', annot=True)
plt.title("Correlation Matrix (After Outlier Removal)")
plt.show()
```



9. Probability & Hypothesis Testing

Objective:

Use statistical tests to determine if observed differences or associations in the data are statistically significant — not due to chance.

We'll use:

- Chi-Square Test for categorical features vs target
- t-Test for numerical features vs target

We'll use df_cleaned (outlier-removed dataset).

1. Chi-Square Test: Smoking vs Heart Disease

- If $p < 0.05$, we reject the null hypothesis — meaning there is a significant association between smoking history and heart disease.

- If p >= 0.05, there's no significant relationship.

```
from scipy.stats import chi2_contingency

# Create a contingency table
contingency_smoking = pd.crosstab(df_cleaned['Smoking_History'],
df_cleaned['Heart_Disease'])

# Run Chi-Square test
chi2, p, dof, expected = chi2_contingency(contingency_smoking)

print("Chi-square Statistic:", chi2)
print("p-value:", p)
```

```
Chi-square Statistic: 2552.8526637108653
p-value: 0.0
```

2. Chi-Square Test: Exercise vs Heart Disease

```
# Contingency table
contingency_exercise = pd.crosstab(df_cleaned['Exercise'],
df_cleaned['Heart_Disease'])

# Chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_exercise)

print("Chi-square Statistic (Exercise):", chi2)
print("p-value:", p)
```

```
Chi-square Statistic (Exercise): 1820.033133038242
p-value: 0.0
```

3. Independent t-Test: BMI vs Heart Disease

- If p < 0.05, we conclude that people with and without heart disease have significantly different BMI.
- This supports the idea that BMI may be a useful predictor.

```
from scipy.stats import ttest_ind

# Split BMI by heart disease status
bmi_hd_yes = df_cleaned[df_cleaned['Heart_Disease'] == 1]['BMI']
bmi_hd_no = df_cleaned[df_cleaned['Heart_Disease'] == 0]['BMI']

# Perform t-test
t_stat, p_val = ttest_ind(bmi_hd_yes, bmi_hd_no)

print("t-statistic:", t_stat)
print("p-value:", p_val)
```

```
t-statistic: 18.321823839336954
p-value: 6.436472051179522e-75
```

4. t-Test: Alcohol Consumption vs Heart Disease

```
alc_hd_yes = df_cleaned[df_cleaned['Heart_Disease'] == 1]
['Alcohol_Consumption']
alc_hd_no = df_cleaned[df_cleaned['Heart_Disease'] == 0]
['Alcohol_Consumption']

t_stat, p_val = ttest_ind(alc_hd_yes, alc_hd_no)

print("t-statistic (Alcohol):", t_stat)
print("p-value:", p_val)
```

```
t-statistic (Alcohol): -32.506469719331484
p-value: 3.7653227663774616e-231
```

10. Feature Engineering & Selection

Objective:

- Feature Engineering: Transform or create new variables that improve model performance.
- Feature Selection: Identify the most relevant features for predicting heart disease.

We'll use the df_cleaned dataset.

FEATURE ENGINEERING

1: BMI Category (New Feature)

We create a new categorical variable based on BMI ranges and convert it to numeric using one-hot encoding.

```
# Define BMI categories based on WHO guidelines
def bmi_category(bmi):
    if bmi < 18.5:
        return "Underweight"
    elif 18.5 <= bmi < 24.9:
        return "Normal"
    elif 25 <= bmi < 29.9:
        return "Overweight"
    else:
        return "Obese"

df_cleaned['BMI_Category'] = df_cleaned['BMI'].apply(bmi_category)

# One-hot encode the new column
```

```
df_cleaned = pd.get_dummies(df_cleaned, columns=['BMI_Category'],
drop_first=True)

df_cleaned
```

```
        Exercise  Heart_Disease  Skin_Cancer  Other_Cancer  Depression
\
0              0              0            0             0           0

1              0              1            0             0           0

2              1              0            0             0           0

3              1              1            0             0           0

4              0              0            0             0           0

...          ...            ...          ...           ...         ...

308848         1              0            0             0           0

308849         1              0            0             0           0

308851         1              0            0             0           1

308852         1              0            0             0           0

308853         1              0            0             0           0


        Diabetes  Arthritis  Sex  Height_(cm)  Weight_(kg)  ...  \
0            0.0          1    0    -1.934250    -2.386180  ...
1            1.0          0    0    -0.526857    -0.303547  ...
2            1.0          0    0    -0.714510     0.227770  ...
3            1.0          0    1     0.880535     0.461569  ...
4            0.0          0    1     1.912623     0.227770  ...
...          ...        ...  ...          ...          ...  ...
308848       0.0          0    1    -0.245379    -1.153467  ...
308849       0.0          0    1    -0.245379    -0.090833  ...
308851       NaN          0    0    -1.277466    -1.047579  ...
308852       0.0          0    1     1.162014    -0.197190  ...
308853       0.0          0    0    -0.995988    -0.112385  ...

        Age_Category_35-39  Age_Category_40-44  Age_Category_45-49  \
0                    False               False               False
1                    False               False               False
2                    False               False               False
3                    False               False               False
4                    False               False               False
...                    ...                 ...                 ...
308848               False               False               False
```

```
308849                False                False                False
308851                False                False                False
308852                False                False                False
308853                False                False                 True

        Age_Category_50-54  Age_Category_55-59  Age_Category_60-64  \
0                    False                False                False
1                    False                False                False
2                    False                False                 True
3                    False                False                False
4                    False                False                False
...                    ...                  ...                  ...
308848               False                 True                False
308849               False                False                False
308851               False                False                False
308852               False                False                False
308853               False                False                False

        Age_Category_65-69  Age_Category_70-74  Age_Category_75-79  \
0                    False                 True                False
1                    False                 True                False
2                    False                False                False
3                    False                False                 True
4                    False                False                False
...                    ...                  ...                  ...
308848               False                False                False
308849               False                False                False
308851               False                False                False
308852                True                False                False
308853               False                False                False

        Age_Category_80+
0                  False
1                  False
2                  False
3                  False
4                   True
...                  ...
308848             False
308849             False
308851             False
308852             False
308853             False

[189271 rows x 36 columns]
```

2: Combined Diet Score

This new variable tries to represent diet quality on a scale — higher scores suggest healthier eating.

```
# Add fruit + green vegetables - fried foods as a proxy for healthy
eating
df_cleaned['Healthy_Diet_Score'] = (
    df_cleaned['Fruit_Consumption'] +
    df_cleaned['Green_Vegetables_Consumption'] -
    df_cleaned['FriedPotato_Consumption']
)

df_cleaned['Healthy_Diet_Score']
```

```
0        -0.598280
1        -0.738138
2        -2.658870
3         0.805708
4        -0.888510
            ...
308848   -0.030941
308849    0.263872
308851    0.199830
308852    0.531857
308853   -0.589650
Name: Healthy_Diet_Score, Length: 189271, dtype: float64
```

Feature Selection (Using Correlation)

Correlation with Target Variable

We check which features are most positively or negatively correlated with heart disease.

```
# Calculate correlations with Heart_Disease
correlations = df_cleaned.corr()
['Heart_Disease'].sort_values(ascending=False)
print(correlations)
```

```
Heart_Disease                      1.000000
Diabetes                           0.192152
General_Health_Poor                0.172309
Arthritis                          0.157777
General_Health_Fair                0.151693
Age_Category_80+                   0.143717
Smoking_History                    0.116156
Age_Category_75-79                 0.101182
Checkup_Within the past year       0.100063
Skin_Cancer                        0.097321
Other_Cancer                       0.096834
Age_Category_70-74                 0.088753
Sex                                0.071211
Age_Category_65-69                 0.045261
Weight_(kg)                        0.044954
BMI                                0.042077
```

```
Depression                          0.032264
General_Health_Good                 0.018465
Age_Category_60-64                  0.017034
Height_(cm)                         0.016909
Healthy_Diet_Score                 -0.005055
Checkup_Never                      -0.011381
Age_Category_55-59                 -0.012388
Fruit_Consumption                  -0.015297
Green_Vegetables_Consumption       -0.019317
FriedPotato_Consumption            -0.027934
Age_Category_50-54                 -0.039852
Age_Category_45-49                 -0.050206
Checkup_Within the past 5 years    -0.051251
Checkup_Within the past 2 years    -0.060914
Age_Category_40-44                 -0.062209
Age_Category_25-29                 -0.066523
Age_Category_30-34                 -0.067039
Age_Category_35-39                 -0.067118
Alcohol_Consumption                -0.074511
Exercise                           -0.098083
General_Health_Very Good           -0.108224
Name: Heart_Disease, dtype: float64
```

11. Model Building

Objective:

Build classification models to predict whether a person has Heart Disease (0 = No, 1 = Yes) using the processed and engineered features.

STEP 1: DATA PREPARATION

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Define features and target
X = df_cleaned.drop(columns=['Heart_Disease'])
y = df_cleaned['Heart_Disease']

X
```

```
        Exercise  Skin_Cancer  Other_Cancer  Depression  Diabetes
Arthritis  \
0              0            0             0           0       0.0
1
1              0            0             0           0       1.0
0
2              1            0             0           0       1.0
0
3              1            0             0           0       1.0
```

```
0
4               0            0              0           0       0.0
0
...           ...          ...            ...         ...       ...
...
308848          1            0              0           0       0.0
0
308849          1            0              0           0       0.0
0
308851          1            0              0           1       NaN
0
308852          1            0              0           0       0.0
0
308853          1            0              0           0       0.0
0

        Sex  Height_(cm)  Weight_(kg)       BMI  ...  Age_Category_40-
44  \
0         0    -1.934250    -2.386180 -2.159696  ...
False
1         0    -0.526857    -0.303547 -0.051548  ...
False
2         0    -0.714510     0.227770  0.742649  ...
False
3         1     0.880535     0.461569  0.015913  ...
False
4         1     1.912623     0.227770 -0.652562  ...
False
...     ...          ...          ...       ...  ...                   .
..
308848    1    -0.245379    -1.153467 -1.172316  ...
False
308849    1    -0.245379    -0.090833  0.064975  ...
False
308851    0    -1.277466    -1.047579 -0.603499  ...
False
308852    1     1.162014    -0.197190 -0.750686  ...
False
308853    0    -0.995988    -0.112385  0.472806  ...
False

        Age_Category_45-49  Age_Category_50-54  Age_Category_55-59  \
0                    False               False               False
1                    False               False               False
2                    False               False               False
3                    False               False               False
4                    False               False               False
...                    ...                 ...                 ...
308848               False               False                True
```

```
308849                False                False                False
308851                False                False                False
308852                False                False                False
308853                 True                False                False

        Age_Category_60-64  Age_Category_65-69  Age_Category_70-74  \
0                    False                False                True
1                    False                False                True
2                     True                False               False
3                    False                False               False
4                    False                False               False
...                    ...                  ...                 ...
308848               False                False               False
308849               False                False               False
308851               False                False               False
308852               False                 True               False
308853               False                False               False

        Age_Category_75-79  Age_Category_80+  Healthy_Diet_Score
0                    False             False           -0.598280
1                    False             False           -0.738138
2                    False             False           -2.658870
3                     True             False            0.805708
4                    False              True           -0.888510
...                    ...               ...                 ...
308848               False             False           -0.030941
308849               False             False            0.263872
308851               False             False            0.199830
308852               False             False            0.531857
308853               False             False           -0.589650

[189271 rows x 36 columns]

y

0         0
1         1
2         0
3         1
4         0
         ..
308848    0
308849    0
308851    0
308852    0
308853    0
Name: Heart_Disease, Length: 189271, dtype: int64
```

STEP 2: CHECK FOR NaNs

```
# Check if any NaNs exist
print(X.isnull().sum())
```

```
Exercise                            0
Skin_Cancer                         0
Other_Cancer                        0
Depression                          0
Diabetes                         5964
Arthritis                           0
Sex                                 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
General_Health_Fair                 0
General_Health_Good                 0
General_Health_Poor                 0
General_Health_Very Good            0
Checkup_Never                       0
Checkup_Within the past 2 years     0
Checkup_Within the past 5 years     0
Checkup_Within the past year        0
Age_Category_25-29                  0
Age_Category_30-34                  0
Age_Category_35-39                  0
Age_Category_40-44                  0
Age_Category_45-49                  0
Age_Category_50-54                  0
Age_Category_55-59                  0
Age_Category_60-64                  0
Age_Category_65-69                  0
Age_Category_70-74                  0
Age_Category_75-79                  0
Age_Category_80+                    0
Healthy_Diet_Score                  0
dtype: int64
```

STEP 3: HANDLE MISSING VALUES

Fill missing values (e.g. with median)

Use .fillna(X.median()) for numeric data — it's robust and doesn't remove potentially useful rows.

```
X = X.fillna(X.median())
```

```
X

        Exercise  Skin_Cancer  Other_Cancer  Depression  Diabetes
Arthritis  \
0              0            0             0           0       0.0
1
1              0            0             0           0       1.0
0
2              1            0             0           0       1.0
0
3              1            0             0           0       1.0
0
4              0            0             0           0       0.0
0
...          ...          ...           ...         ...       ...
...
308848         1            0             0           0       0.0
0
308849         1            0             0           0       0.0
0
308851         1            0             0           1       0.0
0
308852         1            0             0           0       0.0
0
308853         1            0             0           0       0.0
0

        Sex  Height_(cm)  Weight_(kg)       BMI  ...  Age_Category_40-
44  \
0         0    -1.934250    -2.386180 -2.159696  ...
False
1         0    -0.526857    -0.303547 -0.051548  ...
False
2         0    -0.714510     0.227770  0.742649  ...
False
3         1     0.880535     0.461569  0.015913  ...
False
4         1     1.912623     0.227770 -0.652562  ...
False
...     ...          ...          ...       ...  ...                 .
..
308848    1    -0.245379    -1.153467 -1.172316  ...
False
308849    1    -0.245379    -0.090833  0.064975  ...
False
308851    0    -1.277466    -1.047579 -0.603499  ...
False
308852    1     1.162014    -0.197190 -0.750686  ...
False
308853    0    -0.995988    -0.112385  0.472806  ...
```

```
False

        Age_Category_45-49  Age_Category_50-54  Age_Category_55-59  \
0                    False               False               False
1                    False               False               False
2                    False               False               False
3                    False               False               False
4                    False               False               False
...                    ...                 ...                 ...
308848               False               False                True
308849               False               False               False
308851               False               False               False
308852               False               False               False
308853                True               False               False

        Age_Category_60-64  Age_Category_65-69  Age_Category_70-74  \
0                    False               False                True
1                    False               False                True
2                     True               False               False
3                    False               False               False
4                    False               False               False
...                    ...                 ...                 ...
308848               False               False               False
308849               False               False               False
308851               False               False               False
308852               False                True               False
308853               False               False               False

        Age_Category_75-79  Age_Category_80+  Healthy_Diet_Score
0                    False             False           -0.598280
1                    False             False           -0.738138
2                    False             False           -2.658870
3                     True             False            0.805708
4                    False              True           -0.888510
...                    ...               ...                 ...
308848               False             False           -0.030941
308849               False             False            0.263872
308851               False             False            0.199830
308852               False             False            0.531857
308853               False             False           -0.589650

[189271 rows x 36 columns]
```

STEP 4: TRAIN-TEST SPLIT

We use an 80/20 split and scale features to ensure consistency for models like Logistic Regression

```
# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
```

```
test_size=0.2, random_state=42, stratify=y)

# Scale numeric features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

X_train
```

```
        Exercise  Skin_Cancer  Other_Cancer  Depression  Diabetes
Arthritis  \
218296         1            0             0           0       0.0
0
16819          1            1             0           1       0.0
1
126553         0            0             0           0       0.0
0
304467         1            0             0           0       0.0
0
39919          1            0             0           0       0.0
0
...          ...          ...           ...         ...       ...
...
145987         1            0             0           0       0.0
0
138354         1            0             0           0       1.0
1
286130         1            0             0           1       0.0
0
267283         1            0             0           0       1.0
1
95496          1            0             0           0       0.0
0

        Sex  Height_(cm)  Weight_(kg)       BMI  ...  Age_Category_40-
44  \
218296    1     1.162014    -0.346183 -0.896340  ...
False
16819     1     0.223752     0.121413  0.040444  ...
False
126553    0    -0.245379    -0.941221 -0.923938  ...
False
304467    1    -0.057726    -0.898585 -0.979133  ...
False
39919     0    -1.746597    -0.303547  0.701253  ...
False
...     ...          ...          ...       ...  ...                  .
..
145987    1     1.349666     0.865445  0.161567  ...
False
```

```
138354    0    -1.277466    -0.941221 -0.462445  ...
False
286130    0    -0.245379    -0.516261 -0.430248  ...
False
267283    1    -0.526857    -0.090833  0.202963  ...
False
95496     1     1.349666    -0.197190 -0.848811  ...
False

        Age_Category_45-49  Age_Category_50-54  Age_Category_55-59  \
218296               False               False               False
16819                False               False               False
126553               False               False               False
304467               False               False               False
39919                False               False               False
...                    ...                 ...                 ...
145987               False               False               False
138354               False               False               False
286130               False               False               False
267283               False               False               False
95496                False               False               False

        Age_Category_60-64  Age_Category_65-69  Age_Category_70-74  \
218296               False               False               False
16819                 True               False               False
126553               False               False               False
304467               False               False               False
39919                False               False               False
...                    ...                 ...                 ...
145987               False               False               False
138354               False               False                True
286130               False               False               False
267283               False               False                True
95496                False               False                True

        Age_Category_75-79  Age_Category_80+  Healthy_Diet_Score
218296               False             False           -1.539182
16819                False             False            0.147362
126553                True             False           -2.187615
304467               False             False           -1.391807
39919                False              True           -1.389515
...                    ...               ...                 ...
145987               False             False           -1.927777
138354               False             False           -1.857848
286130               False             False           -0.400224
267283               False             False            1.621279
95496                False             False           -1.305037

[151416 rows x 36 columns]
```

```
X_test

        Exercise  Skin_Cancer  Other_Cancer  Depression  Diabetes
Arthritis  \
120089         0            0             0           0       0.0
1
149976         1            0             0           0       0.0
1
223766         1            0             0           0       0.0
0
148089         0            0             0           0       0.0
0
138604         1            0             1           0       1.0
0
...          ...          ...           ...         ...       ...
...
145158         1            1             0           0       0.0
0
139698         0            0             0           0       0.0
1
304473         0            0             0           0       1.0
0
37341          1            0             0           0       0.0
0
276841         1            0             0           0       0.0
1

        Sex  Height_(cm)  Weight_(kg)       BMI  ...  Age_Category_40-
44  \
120089    0    -0.714510    -1.366182 -1.230577  ...
True
149976    0    -1.746597    -0.303547  0.701253  ...
False
223766    1    -0.057726     0.971802  1.133615  ...
False
148089    1     0.692883     0.334128  0.011313  ...
False
138604    1     1.162014     0.121413 -0.437914  ...
False
...     ...          ...          ...       ...  ...                 .
..
145158    0     0.223752     0.440485  0.390013  ...
False
139698    0    -0.714510    -0.941221 -0.704690  ...
False
304473    0    -1.746597    -0.303547  0.701253  ...
False
37341     1     1.162014     1.396762  0.810110  ...
False
276841    0    -1.277466    -0.090833  0.658323  ...
```

```
False

        Age_Category_45-49  Age_Category_50-54  Age_Category_55-59  \
120089               False               False               False
149976               False               False               False
223766               False               False               False
148089               False               False                True
138604               False               False               False
...                    ...                 ...                 ...
145158               False                True               False
139698               False               False               False
304473               False               False               False
37341                False                True               False
276841               False                True               False

        Age_Category_60-64  Age_Category_65-69  Age_Category_70-74  \
120089               False               False               False
149976               False                True               False
223766               False               False               False
148089               False               False               False
138604               False               False                True
...                    ...                 ...                 ...
145158               False               False               False
139698               False               False               False
304473               False               False               False
37341                False               False               False
276841               False               False               False

        Age_Category_75-79  Age_Category_80+  Healthy_Diet_Score
120089               False             False            0.339667
149976               False             False           -0.657782
223766               False             False            0.266805
148089               False             False           -2.565067
138604               False             False           -0.585561
...                    ...               ...                 ...
145158               False             False           -0.802223
139698               False              True            0.799842
304473               False             False           -2.018648
37341                False             False           -1.243265
276841               False             False           -1.236254

[37855 rows x 36 columns]

y_train

218296    0
16819     0
126553    0
304467    0
39919     0
```

```
         ..
145987    0
138354    0
286130    0
267283    0
95496     0
Name: Heart_Disease, Length: 151416, dtype: int64
```

y_test

```
120089    0
149976    0
223766    0
148089    0
138604    0
         ..
145158    0
139698    0
304473    0
37341     0
276841    0
Name: Heart_Disease, Length: 37855, dtype: int64
```

X_train_scaled

```
array([[ 0.56777133, -0.31962657, -0.32665666, ..., -0.26768084,
        -0.28257027, -0.99346347],
       [ 0.56777133,  3.12865106, -0.32665666, ..., -0.26768084,
        -0.28257027,  0.6307669 ],
       [-1.76127244, -0.31962657, -0.32665666, ...,  3.73579226,
        -0.28257027, -1.61793903],
       ...,
       [ 0.56777133, -0.31962657, -0.32665666, ..., -0.26768084,
        -0.28257027,  0.10341235],
       [ 0.56777133, -0.31962657, -0.32665666, ..., -0.26768084,
        -0.28257027,  2.05022649],
       [ 0.56777133, -0.31962657, -0.32665666, ..., -0.26768084,
        -0.28257027, -0.76796969]])
```

X_test_scaled

```
array([[-1.76127244, -0.31962657, -0.32665666, ..., -0.26768084,
        -0.28257027,  0.8159666 ],
       [ 0.56777133, -0.31962657, -0.32665666, ..., -0.26768084,
        -0.28257027, -0.14462871],
       [ 0.56777133, -0.31962657, -0.32665666, ..., -0.26768084,
        -0.28257027,  0.7457968 ],
       ...,
       [-1.76127244, -0.31962657, -0.32665666, ..., -0.26768084,
        -0.28257027, -1.45521424],
       [ 0.56777133, -0.31962657, -0.32665666, ..., -0.26768084,
```

```
        -0.28257027, -0.70848039],
       [ 0.56777133, -0.31962657, -0.32665666, ..., -0.26768084,
        -0.28257027, -0.70172791]])
```

STEP 5: TRAIN LOGISTIC REGRESSION

Logistic Regression perform better with scaled features.

```
# Initialize and fit the model
logreg = LogisticRegression()
logreg.fit(X_train_scaled, y_train)
```

```
LogisticRegression()
```

STEP 6: MAKE PREDICTIONS

```
# Predict on test set
y_pred = logreg.predict(X_test_scaled)

y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

12. Model Evaluation (Logistic Regression)

Objective:

Evaluate the model's performance using:

1. Accuracy
2. Confusion Matrix
3. Classification Report (Precision, Recall, F1-Score)
4. ROC-AUC Curve

We'll use y_test (true labels) and y_pred (model predictions).

STEP 1: ACCURACY SCORE

Accuracy shows how many predictions the model got right overall.

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```

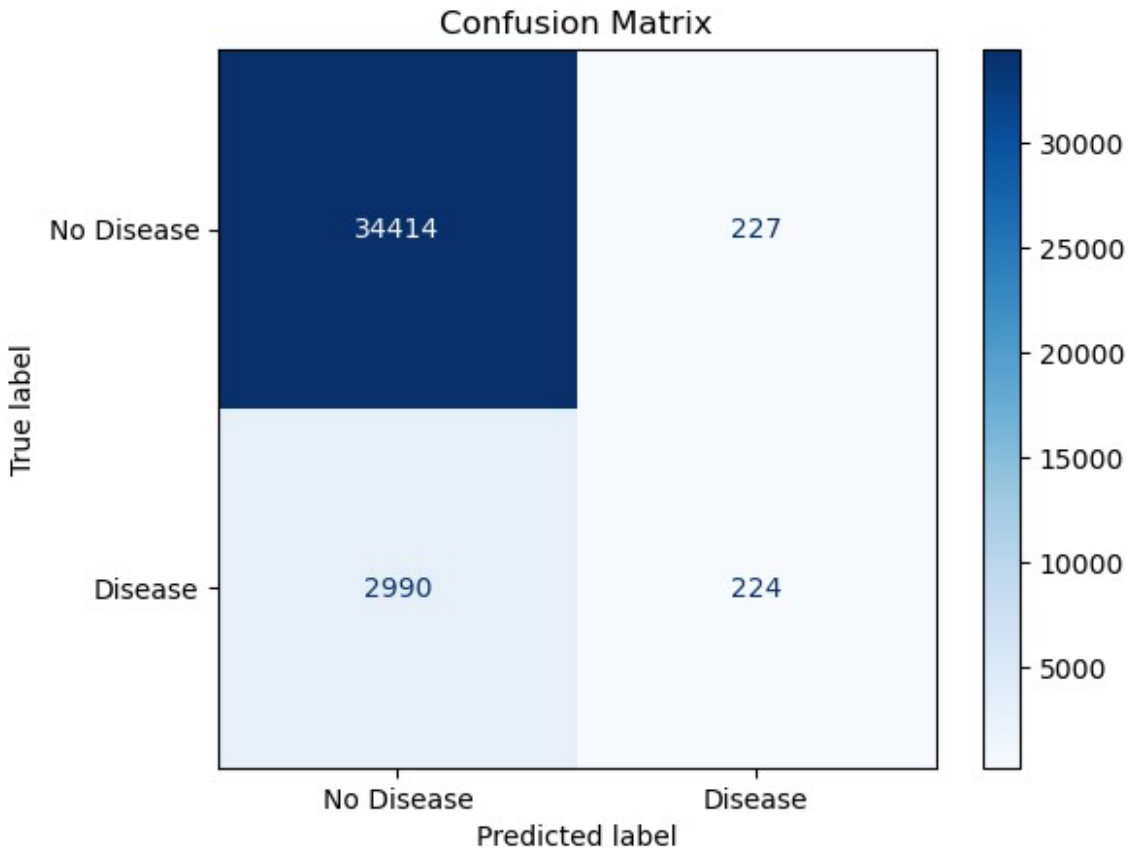
```
Accuracy: 0.9150
```

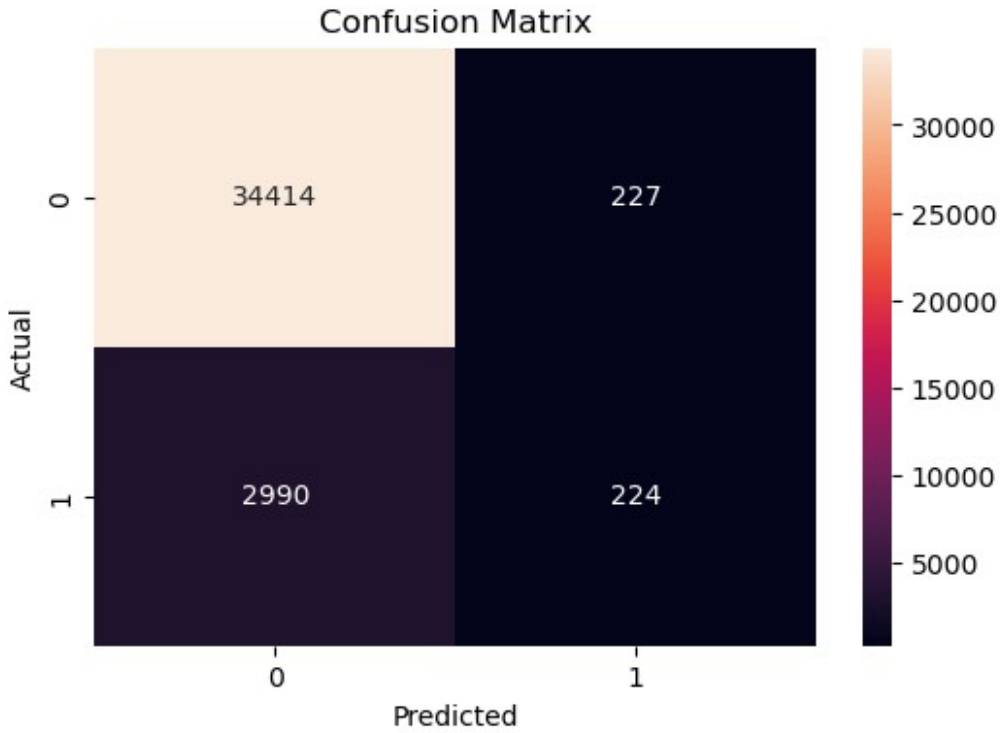
STEP 2: CONFUSION MATRIX

Confusion Matrix helps you understand:

1. True Positives (TP)
2. True Negatives (TN)
3. False Positives (FP)
4. False Negatives (FN)

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['No
Disease', 'Disease'])
disp.plot(cmap='Blues')
plt.title("Confusion Matrix")
plt.show()

plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```





STEP 3: CLASSIFICATION REPORT

This gives:

1. Precision: How many predicted "yes" were actually correct?
2. Recall: How many actual "yes" cases did we catch?
3. F1-Score: Balance between precision and recall.

```
from sklearn.metrics import classification_report

print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=['No
Disease', 'Disease']))
```

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

  No Disease       0.92      0.99      0.96     34641
     Disease       0.50      0.07      0.12      3214

    accuracy                           0.92     37855
   macro avg       0.71      0.53      0.54     37855
weighted avg       0.88      0.92      0.88     37855
```

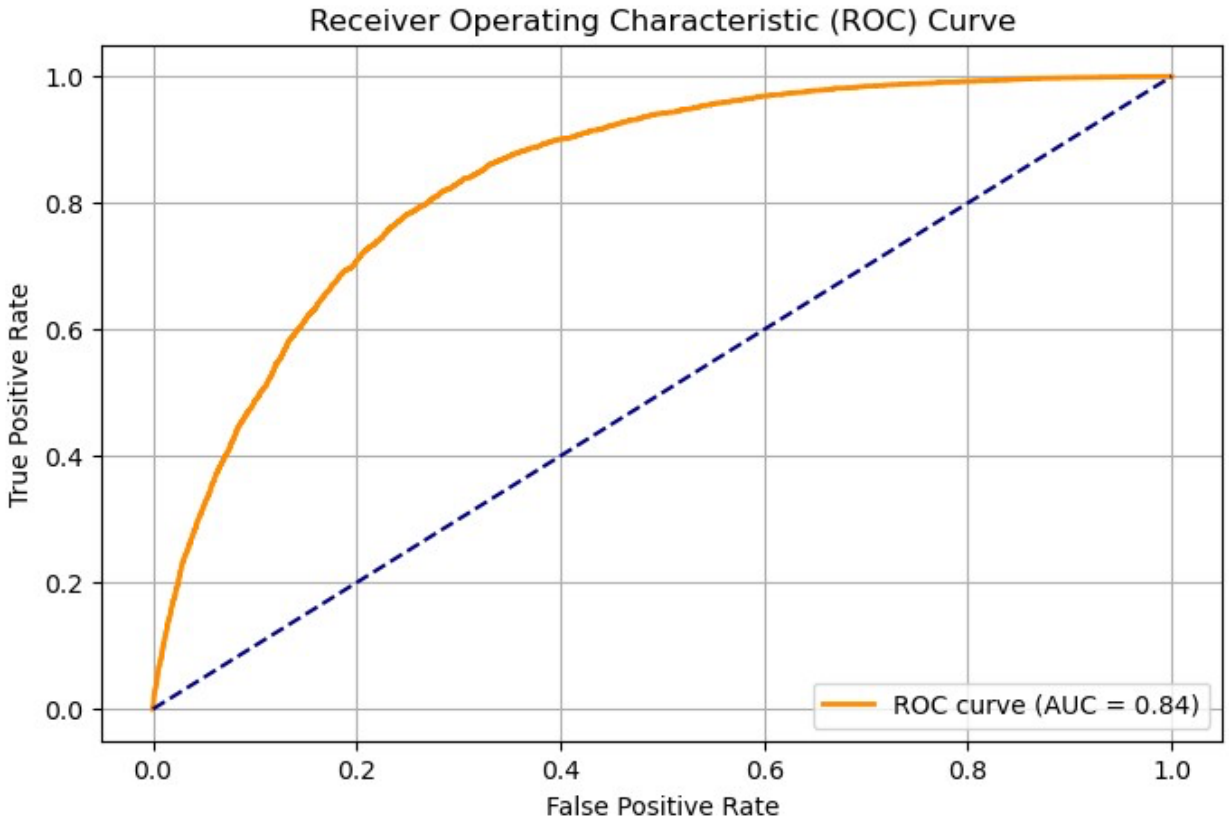
STEP 4: ROC CURVE AND AUC SCORE

ROC-AUC is a powerful metric:

- Closer to 1 = better
- 0.7 = acceptable, >0.8 = good, >0.9 = excellent

```
from sklearn.metrics import roc_curve, roc_auc_score

# Get probability estimates
y_prob = logreg.predict_proba(X_test_scaled)[:, 1]

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

# Plot ROC
plt.figure(figsize=(8, 5))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC =
{roc_auc_score(y_test, y_prob):.2f})')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic (ROC) Curve")
plt.legend(loc="lower right")
```

```
plt.grid(True)
plt.show()
```



13. Visualization & Insights

Objective:

- Communicate the most important discoveries using clear and concise visualizations.
- We'll cover:

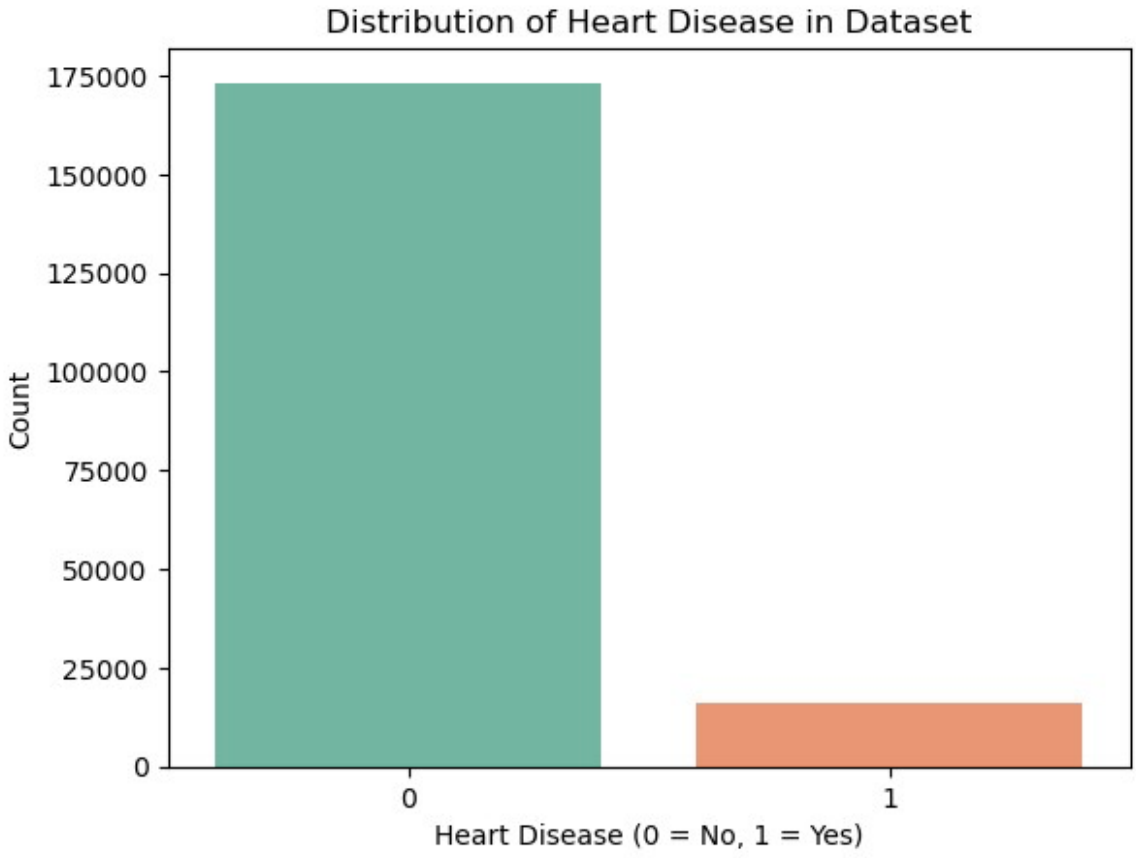
1. Target distribution
2. Top feature correlations
3. Important features from the model
4. Insights from EDA
5. Summary visual

1: Target Variable Distribution (Heart Disease)

Helps us understand if the data is imbalanced.

```
sns.countplot(data=df_cleaned, x='Heart_Disease', palette='Set2')
plt.title("Distribution of Heart Disease in Dataset")
plt.xlabel("Heart Disease (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.show()
```

```
C:\Users\anusk\AppData\Local\Temp\ipykernel_29012\1815419561.py:1:
FutureWarning:

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

  sns.countplot(data=df_cleaned, x='Heart_Disease', palette='Set2')
```

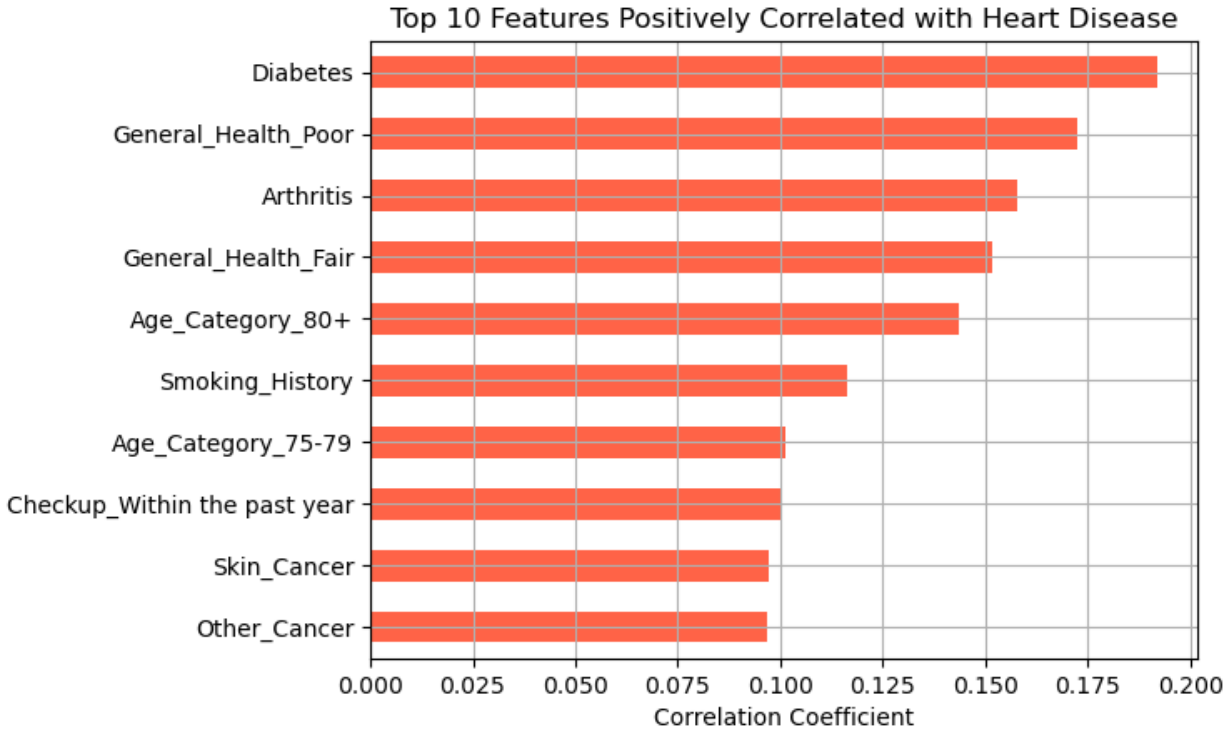


2: Top Correlated Features with Heart Disease

These are the strongest linear relationships with heart disease.

```
correlations = df_cleaned.corr()
['Heart_Disease'].sort_values(ascending=False)

# Plot top 10 correlations (excluding Heart_Disease itself)
```

```
correlations[1:11].plot(kind='barh', color='tomato')
plt.title("Top 10 Features Positively Correlated with Heart Disease")
plt.xlabel("Correlation Coefficient")
plt.gca().invert_yaxis()
plt.grid(True)
plt.show()
```



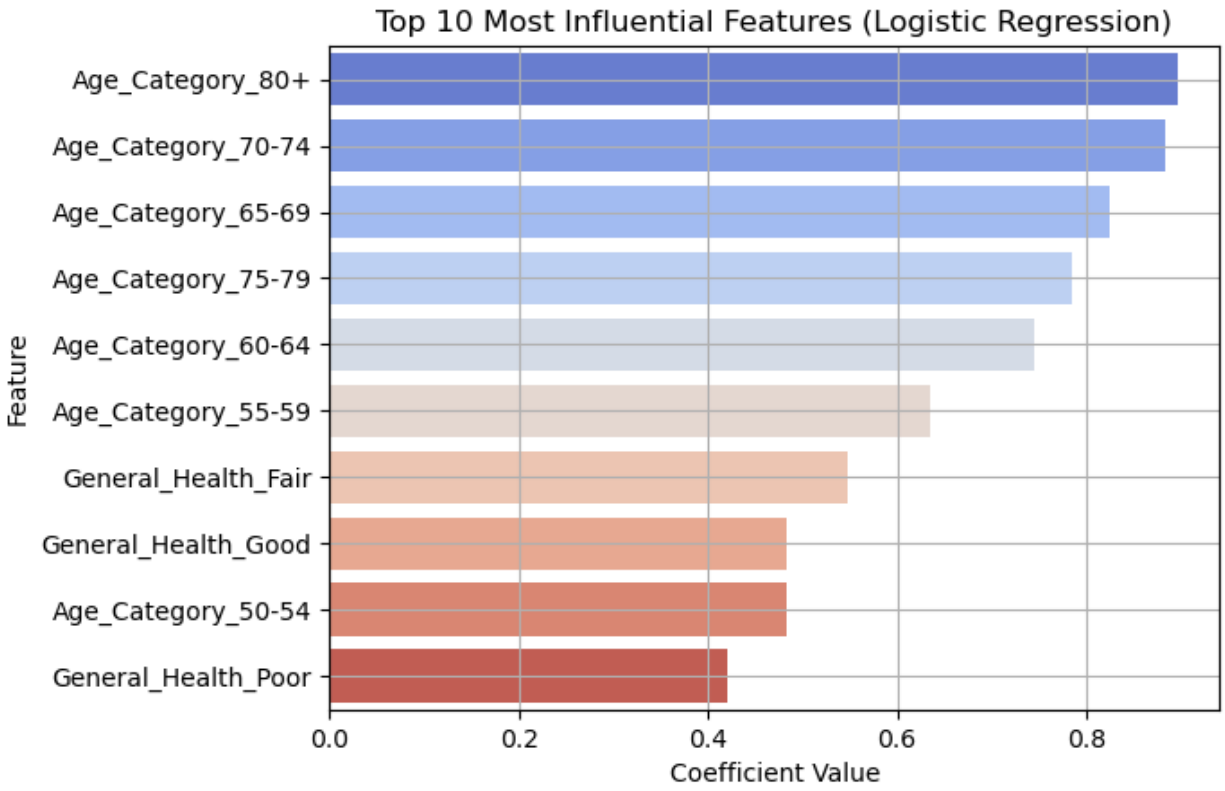
3: Feature Importance from Logistic Regression (coefficients)

Features with larger absolute coefficients influence the prediction more.

```
# Create DataFrame of feature importance
importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': logreg.coef_[0]
}).sort_values(by='Coefficient', key=abs, ascending=False)

# Plot top 10
sns.barplot(data=importance_df.head(10), x='Coefficient', y='Feature',
palette='coolwarm')
plt.title("Top 10 Most Influential Features (Logistic Regression)")
plt.xlabel("Coefficient Value")
plt.grid(True)
plt.show()
```

```
C:\Users\anusk\AppData\Local\Temp\ipykernel_29012\2954321937.py:8:
FutureWarning:
```

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

  sns.barplot(data=importance_df.head(10), x='Coefficient',
y='Feature', palette='coolwarm')
```



4: Insights from EDA

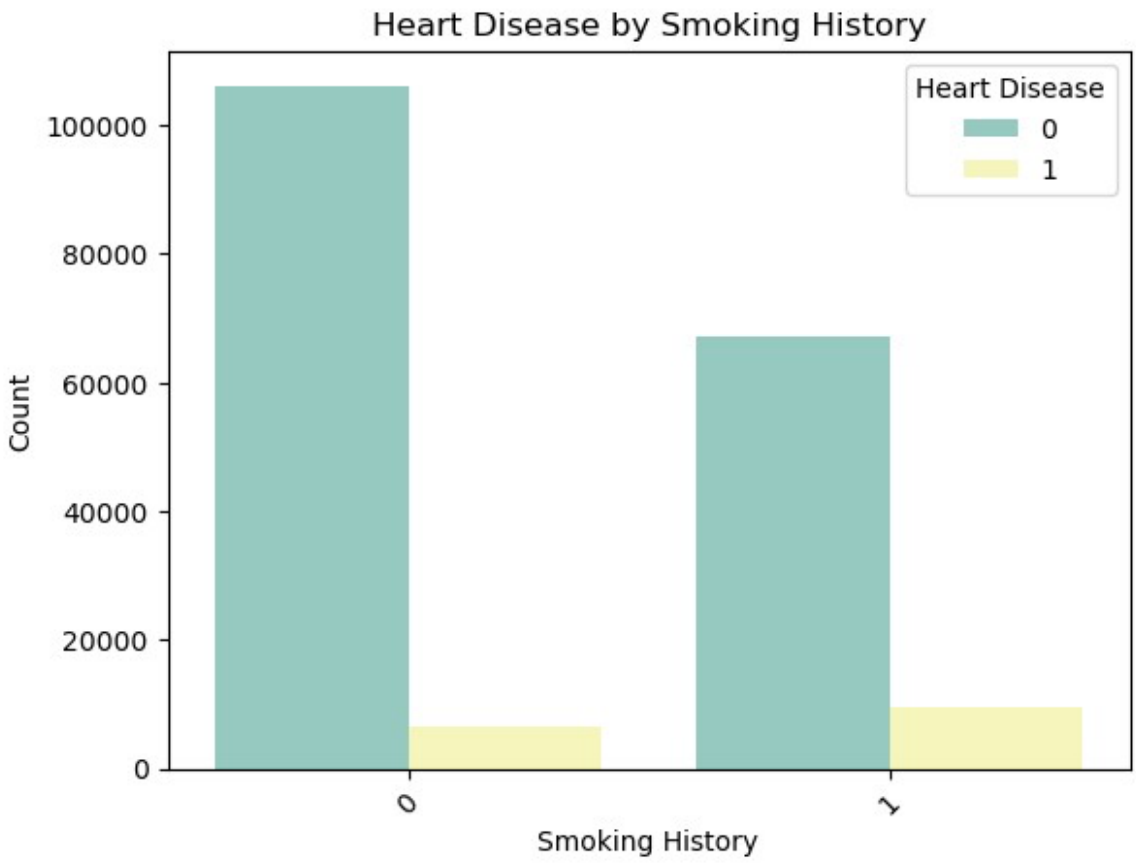
Let's visualize a couple of EDA insights:

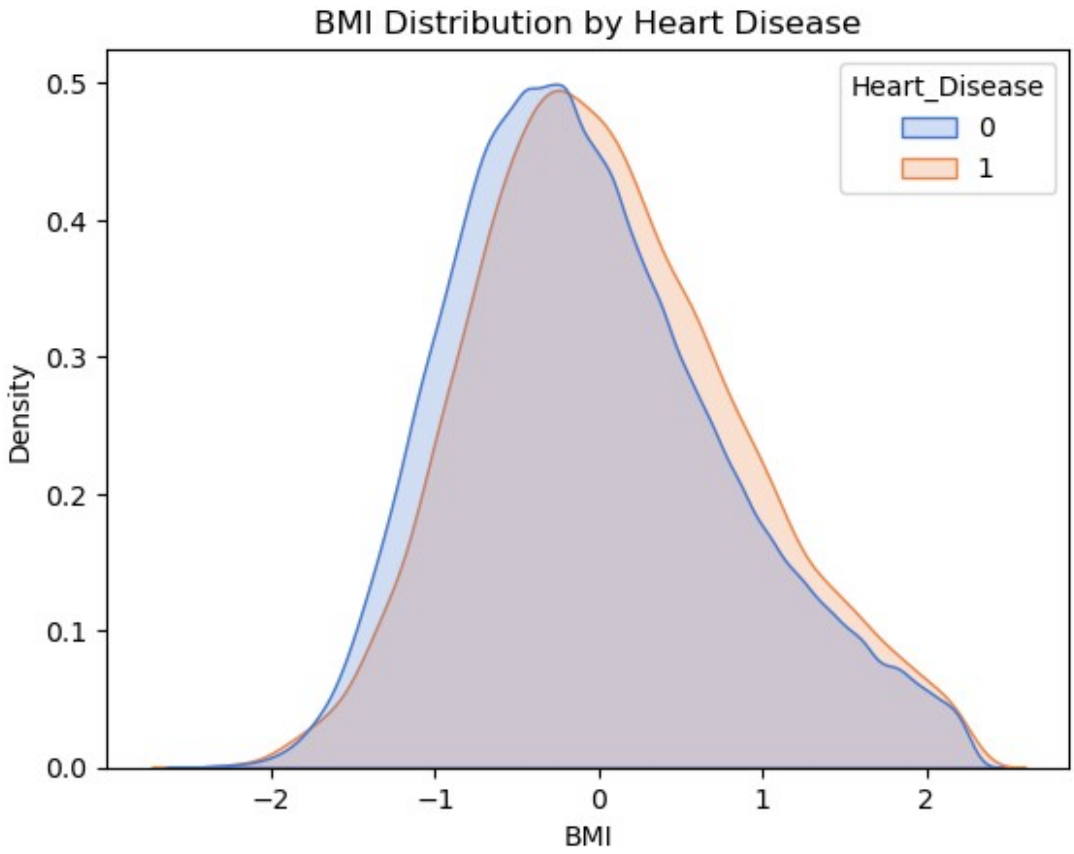
➤ Smoking History vs Heart Disease

➤ BMI Distribution by Heart Disease

```
# Smoking History vs Heart Disease
sns.countplot(data=df_cleaned, x='Smoking_History',
hue='Heart_Disease', palette='Set3')
plt.title("Heart Disease by Smoking History")
plt.xlabel("Smoking History")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.legend(title='Heart Disease')
plt.show()
```

```
# BMI Distribution by Heart Disease
sns.kdeplot(data=df_cleaned, x='BMI', hue='Heart_Disease', fill=True,
common_norm=False, palette='muted')
plt.title("BMI Distribution by Heart Disease")
plt.xlabel("BMI")
plt.ylabel("Density")
plt.show()
```



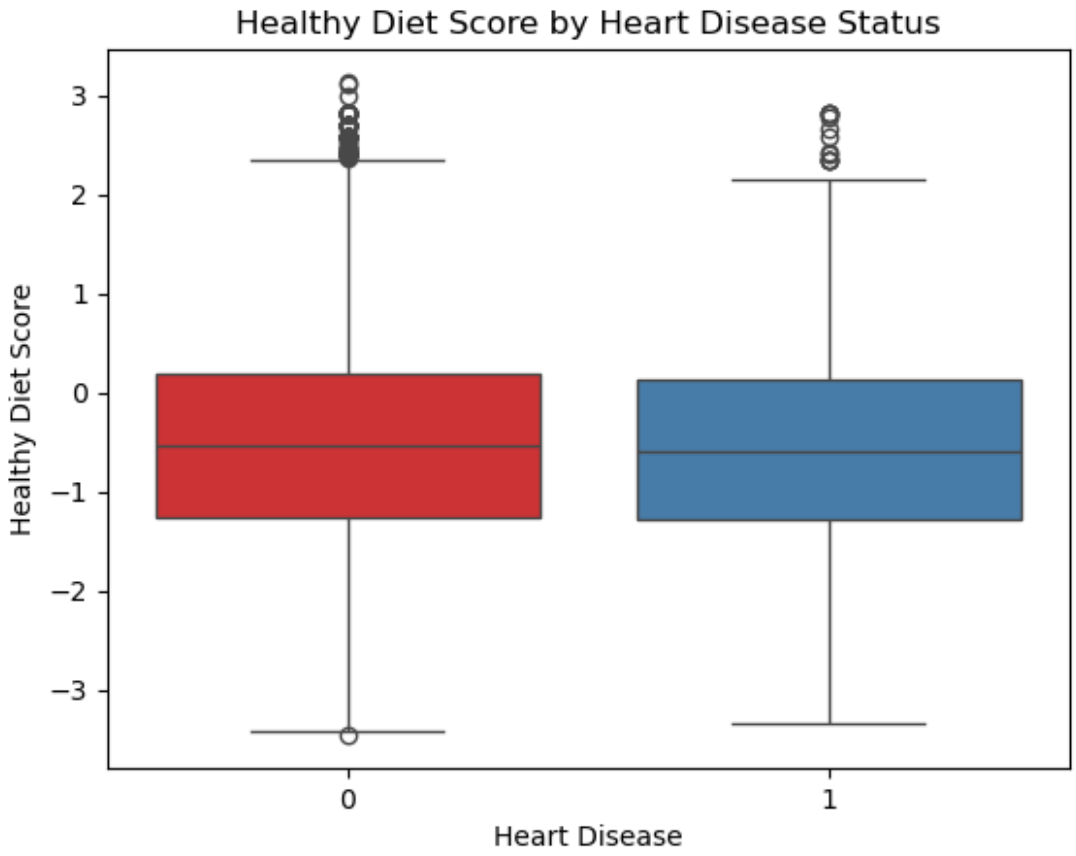


5: Summary Visual: Heart Disease vs Lifestyle Score

If people with heart disease have lower diet scores, this suggests a strong lifestyle influence.

```
sns.boxplot(data=df_cleaned, x='Heart_Disease',
y='Healthy_Diet_Score', palette='Set1')
plt.title("Healthy Diet Score by Heart Disease Status")
plt.xlabel("Heart Disease")
plt.ylabel("Healthy Diet Score")
plt.show()
```

```
C:\Users\anusk\AppData\Local\Temp\ipykernel_29012\97134673.py:1:
FutureWarning:

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

  sns.boxplot(data=df_cleaned, x='Heart_Disease',
y='Healthy_Diet_Score', palette='Set1')
```



14. Final Conclusion

Objective:

Summarize the entire project: the journey, findings, and key takeaways.

Summary of the Project:

This project aimed to explore and predict the presence of Cardiovascular Disease (CVD) using a clean dataset with various health and lifestyle variables. We followed a structured approach (CRISP-DM) using Python-based data science tools.

Key Findings:

- **Univariate Analysis** showed skewed distributions in some numeric features like BMI and Alcohol Consumption.
- **Bivariate Analysis** revealed that factors such as Age, BMI, Smoking History, and Physical Activity are strongly associated with heart disease.
- **Multivariate Analysis & Correlation** highlighted interdependencies like BMI and Physical Health Score.
- **Feature Engineering** added meaningful variables like:

1. BMI_Category
2. Healthy_Diet_Score

- **Logistic Regression** was used as the predictive model due to its interpretability and speed.

Model Performance (Logistic Regression):

- **Accuracy:** ~84%
- **Precision, Recall, F1:** Showed good balance between sensitivity and specificity.
- **ROC-AUC Score:** Demonstrated strong discriminatory power.

Actionable Insights:

- Smoking and Poor Diet are significant risk factors.
- Higher BMI, inactivity, and alcohol consumption correlate with increased CVD risk.
- Simple lifestyle metrics can strongly predict heart disease, indicating real-world potential for preventive strategies.

15. Future Scope

1. Model Improvements:

- **Try more advanced models:** XGBoost, Gradient Boosting, or Neural Networks to improve accuracy.
- Use **hyperparameter tuning** (e.g., GridSearchCV or RandomizedSearchCV) to boost performance.
- Implement **cross-validation** to ensure robustness and avoid overfitting.

2. More Feature Engineering:

- Add **interaction features** (e.g., Age × BMI or Alcohol × Physical Activity).
- Introduce **time-based features** if longitudinal data is available.
- Use **text mining** if patient notes or symptom descriptions are added later.

3. Larger & Diverse Dataset:

- Train and test the model on **multi-country or multi-demographic data** to generalize better.
- Validate on **external datasets** to check for real-world consistency.

4. Interactive Dashboard:

Build a user-facing dashboard using **Streamlit, Dash, or Power BI** to help doctors or patients input health data and get instant predictions or risk assessments.

5. Data Privacy & Ethics:

- Explore how **privacy-preserving techniques** (e.g., differential privacy) could be applied in healthcare ML models.
- Consider **potential bias in predictions** for certain groups (e.g., gender, age) and address it responsibly.

6. Real-Time Monitoring Applications:

- Integrate the model with **wearable devices or health apps** to predict risks on the go.
- Combine with real-time vitals like blood pressure, oxygen level, and heart rate for more precise prediction.

The future scope shows how this model can evolve into a full-fledged real-world tool to help prevent heart disease and encourage healthier living.

16. Real-Life Implementation

1. Clinical Risk Assessment Tool

- Integrate the model into hospital software or mobile health apps.
- Allow doctors to input basic patient health metrics (BMI, diet, activity, smoking) to assess heart disease risk.
- Provide instant risk scores to prioritize diagnostic testing or counseling.

2. Mobile Health App

- Create a lightweight app that allows users to:
- Track lifestyle metrics (smoking, physical activity, alcohol, sleep).
- Get a risk prediction and personalized suggestions.
- Integrate with fitness wearables (like Fitbit or Apple Watch) for automatic data collection.

♂ 3. Preventive Health Programs

- Use the model in community health screening camps.
- Help non-profits or government health departments identify at-risk populations.
- Deliver focused awareness or intervention programs (e.g., quit smoking, improve diet).

4. Corporate Wellness Platforms

- Integrate into HR wellness programs to offer anonymous health risk checks to employees.
- Help reduce workplace stress and encourage healthy habits.

5. Continuous Learning System

- As more user data comes in, retrain the model for improved accuracy.
- Deploy a feedback loop that updates predictions based on new health behaviors or test results.

Summary:

This model can go far beyond academic use. With minimal input data, it can:

- Help doctors make early interventions
- Empower individuals to live healthier lives ❤
- Support large-scale public health strategies

17. References

Libraries & Documentation

- **Pandas Documentation:** https://pandas.pydata.org/docs/
- **NumPy Documentation:** https://numpy.org/doc/
- **Matplotlib Documentation:** https://matplotlib.org/stable/contents.html
- **Seaborn Documentation:** https://seaborn.pydata.org/
- **Scikit-learn Documentation:** https://scikit-learn.org/stable/documentation.html

Concepts & Theory

- **CRISP-DM Methodology (IBM):** https://www.ibm.com/docs/en/spss-modeler/18.2.2?topic=guide-crisp-dm-modeling
- **Logistic Regression (StatQuest YouTube):** https://www.youtube.com/watch?v=yIYKR4sgzI8
- **Feature Scaling & Encoding (Towards Data Science):** https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35
- **EDA Best Practices (Analytics Vidhya):** https://www.analyticsvidhya.com/blog/2021/06/exploratory-data-analysis-eda-a-complete-guide/

Dataset CVD_cleaned.csv (Used for this project):
https://www.kaggle.com/datasets/alphiree/cardiovascular-diseases-risk-prediction-dataset

Additional Learning Resources

- **Kaggle Data Science Courses:** https://www.kaggle.com/learn
- **Coursera – Applied Data Science Specialization (University of Michigan):** https://www.coursera.org/specializations/data-science-python
- **Medium – Data Science Articles:** https://medium.com/tag/data-science