A CROSS-SECTIONAL STUDY OF ANALYSIS OF CARDIOVASCULAR DISEASE BASED ON HEALTH INDICATORS

Deepishka Pemmasani [1], Daniel Adepoju [1], Poorva Reddy Vanga [1], Raajitha Muthyala [1], Surya Tejaswi Mallidi [1]

[1]Indiana University-Purdue University, Indianapolis, IN 46202, USA,

dpemmasa@iu.edu, Dadepoju@iu.edu, Vanredd@iu.edu, rmuthya@iu.edu, smallid@iu.edu

Abstract: Cardiovascular diseases (CVD) pose a significant public health concern globally, contributing to a substantial burden of morbidity and mortality. This cross-sectional analysis aims to identify significant health indicators associated with cardiovascular disease and develop predictive machine learning models. The study focused on understanding the prevalence and determinants of CVD risk factors to facilitate the development of effective interventions for reducing the burden of CVD. The dataset, obtained from Kaggle, included variables such as age, gender, blood pressure, cholesterol level, smoking, alcohol consumption, and physical activity. The logistic regression model showed significant associations between the included health indicators and the presence of cardiovascular disease. The positive coefficients for "systolic BP", "Diastolic BP", and "cholesterol" suggested a higher likelihood of cardiovascular disease, while the negative coefficients for "gluc", "smoke", "alco", and "active" indicated a lower likelihood of cardiovascular disease. The findings underscored the importance of these health indicators in assessing the risk of cardiovascular disease, providing valuable insights for healthcare professionals and policymakers to target specific risk factors and improve cardiovascular health outcomes.

Keywords: Cardiovascular disease, Health indicators, Cross-sectional study, Data analysis, Machine Learning

1 Project Scope

1.1 Introduction

Cardiovascular diseases (CVD) are a significant public health concern, contributing to a substantial burden of morbidity and mortality worldwide. Risk factors for CVD include smoking, diabetes, obesity, and hypertension, which account for about 50% of the pathogenesis of CVD. Treatment of these risk factors has been shown to reduce the risk of future cardiac events (Acevedo et al., 2021). Identifying significant health indicators associated with cardiovascular disease can help in the prevention, diagnosis, and treatment of the disease. Understanding the prevalence and determinants of CVD risk factors is crucial for developing effective interventions to reduce the burden of CVD.

1.2 Aim

Our aim is to identify significant health indicators associated with cardiovascular disease and to develop predictive machine learning models to predict the presence of cardiovascular disease based on these indicators. To address this aim, we posed three research questions:

1. What are the significant health indicators associated with cardiovascular disease?
2. How do various health indicators relate to the presence of cardiovascular disease?
3. Can predictive machine learning models be developed to accurately predict the presence of cardiovascular disease based on health indicators?

Based on our research questions, our two hypotheses are below:

- Null Hypothesis – There is no significant relationship between the health parameters (age, gender, blood pressure, BMI, cholesterol level, smoking, alcohol consumption, physical activity etc.) and the risk of cardiovascular disease (CVD) in the study population.

- Alternate Hypothesis - There is a significant relationship between the health parameters (age, gender, blood pressure, BMI, cholesterol level, smoking, alcohol consumption, physical activity etc.) and the risk of cardiovascular disease (CVD) in the study population.

1.3 Purpose

To Conduct a cross-sectional study to analyze the health screening dataset and to understand the relationship between various health indicators and the presence of cardiovascular disease.

2 Methodology

2.1 Steps of the Project

The main focus of our project will consist of comparing data of risk factors like systolic and diastolic blood pressure, age, smoking, alcoholism, physical activity, cholesterol and glucose levels etc. with the cardiovascular disease incidence rate using database technologies such as SQL and Python. The tools we have used are Python Jupyter notebook, phpMyAdmin, and Tableau 10.5 Beta version.

The methodology of our project involves 4 stages, which will be explained in more depth later on in the report.

1. Data Collection
2. Data Extraction and Storage
3. Data Analysis
4. Data Visualization

2.2 Original Team Members and Responsibilities

Our team has a diverse set of skills and backgrounds. Below was the original list of team members we had for the project, as well as the responsibilities we agreed on at the beginning of the project.

Name	Background	Responsibilities
Daniel Adepoju	MD, worked as a medical officer with World Health Organization Nigeria site. Beginner in SQL and Python.	SQL, Python Coding, Machine Learning integration, and presentation. Help with other responsibilities within the project.
Deepishka Pemmasani	PharmD, worked as Pharmacovigilance Operations specialist at IQVIA Pvt Ltd. Beginner in Python and SQL.	Data Visualization Data Analysis Python coding Machine Learning integration
Poorva Reddy Vanga	Bachelors in Dental Surgery. Beginner in SQL and Python	Data cleaning, Data Visualization, Data Analysis
Raajitha Muthyala	Bachelors in Pharmacy. Beginner in SQL and Python	SQL, Data collection Data cleaning
Surya Tejaswi Mallidi	Pharm D. Experienced in SQL and beginner in Python.	Data Cleaning, Data pre-processing, Exploratory Data Analysis

Table. 1. Original Responsibilities of Team Members for This Project

2.3 Project Challenges

Working in a team will make everyone face challenges. The foremost challenge which we faced was the communication issues in the starting stages of the project, where few people could not make up to the meetings where important aspects were discussed and as a result the work was being done by the remaining members, thus leading to increase in work burden and buildup of communication gap. Gradually, we overcame this challenge by making strict schedules for meetings and online meetings to pass on the information to the remaining candidates.

Another challenge was the removal of outliers. We had to select the central tendency method in order to handle the outliers as our data was not normally distributed. It was a challenge for us as most of the team members were from a science background and were not much experienced in the field of data analysis.

The next challenge was coding in python. Staring from the descriptive statistics till selecting the correct machine learning model for our data set included complex codes. We struggled very much to get the correct codes as most of the time we ended up getting errors. However, when compared to the complexity of the above challenges, dealing with tableau was less complex and with lots of practice, we could complete the coding part.

When setting up online meetings to update the information to the candidates who could not attend the in-person meetings was little troublesome as we did not have enough time to coney the information because the entire meeting ended up in just one hour and we had to wait for another couple of minutes to restart the meeting or start a new one.

Besides all these challenges, our team put in their efforts completely and strived hard to take the project to the finish line.

3 Data Collection

The dataset was taken from the Kaggle https://www.kaggle.com/datasets/drateendrajha/health-screening-data. This dataset provides variables ID, age, gender, height, weight, and health indicators of ap_hi (systolic blood pressure), ap_lo (diastolic blood pressure), cholesterol (levels), gluc (glucose levels), smoke (smoking), cardio (cardiovascular disease), ageinyr (age in year), BMI (body mass index), BMICat (BMI category) and Age Group.

4 Data Extraction and Storage

4.1 Data Extraction

We have imported the dataset into pandas dataframe and assigned it to variable df. We used shape function to read the number of columns and rows. Info function is used to determine the total number values in each column with their data types.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#load data
file_path = 'Health Screening Data.csv'
df = pd.read_csv(file_path)
#explore data
print ('Data shape:',df.shape)
print (df.info())
```

```
Data shape: (69960, 18)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69960 entries, 0 to 69959
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Unnamed: 0     69960 non-null  int64
 1   id             69960 non-null  int64
 2   age            69960 non-null  int64
 3   gender         69960 non-null  int64
 4   height         69960 non-null  int64
 5   weight         69960 non-null  float64
 6   ap_hi          69960 non-null  int64
 7   ap_lo          69960 non-null  int64
 8   cholesterol    69960 non-null  int64
 9   gluc           69960 non-null  int64
 10  smoke          69960 non-null  int64
 11  alco           69960 non-null  int64
 12  active         69960 non-null  int64
 13  cardio         69960 non-null  int64
 14  AgeinYr        69960 non-null  int64
 15  BMI            69960 non-null  float64
 16  BMICat         69960 non-null  object
 17  AgeGroup       69960 non-null  object
dtypes: float64(2), int64(14), object(2)
memory usage: 9.6+ MB
None
```

Fig. 1. Dataset information with variable names, total number of values, data-types

Columns with categories -

- Cholesterol : 1 = level 1 cholesterol, 2 = level 2 cholesterol, 3 = level 3 cholesterol
- Gender: 1 = male, 2 = female
- Glu: 1 = level 1 glucose, 2 = level 2 glucose, 3 = level 3 glucose
- Smoke: 1 = smoker, 0 = non-smoker
- Alco: 1 = alcoholic, 0 = non-alcoholic
- Active: 1 = physically active, 0 = physically inactive
- Cardio: 1 = presence of cardiovascular disease, 0 = absence of cardiovascular disease

4.2 Data Cleaning

The dataset did not have any missing values. The columns were renamed for better understanding of the data. The columns BMI Cat and Age Group had non-numerical values and the columns gender, cholesterol, gluc, smoke, alcohol, active, cardio have categorical values.

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

```
Unnamed: 0     0
id             0
age            0
gender         0
height         0
weight         0
ap_hi          0
ap_lo          0
cholesterol    0
gluc           0
smoke          0
alco           0
active         0
cardio         0
AgeinYr        0
BMI            0
BMICat         0
AgeGroup       0
dtype: int64
```

Fig. 2. Total number of missing values in each variable

The columns of 'ap_hi' and 'ap_lo' were renamed to systolic BP and Diastolic BP for better understanding of the data.

```
df.columns

Index(['Unnamed: 0', 'id', 'age', 'gender', 'height', 'weight', 'ap_hi',
       'ap_lo', 'cholesterol', 'gluc', 'smoke', 'alco', 'active', 'cardio',
       'AgeinYr', 'BMI', 'BMICat', 'AgeGroup'],
      dtype='object')

df = df.rename(columns={'ap_hi': 'systolic BP', 'ap_lo': 'Diastolic BP'})
df.columns

Index(['Unnamed: 0', 'id', 'age', 'gender', 'height', 'weight', 'systolic BP',
       'Diastolic BP', 'cholesterol', 'gluc', 'smoke', 'alco', 'active',
       'cardio', 'AgeinYr', 'BMI', 'BMICat', 'AgeGroup'],
      dtype='object')
```

Fig. 3. Variable names of the dataset

```
df.dtypes

Unnamed: 0          int64
id                  int64
age                 int64
gender              int64
height              int64
weight            float64
systolic BP         int64
Diastolic BP        int64
cholesterol         int64
gluc                int64
smoke               int64
alco                int64
active              int64
cardio              int64
AgeinYr             int64
BMI               float64
BMICat             object
AgeGroup           object
dtype: object
```

Fig. 4. Data types of values in each variable

5. Exploratory data analysis:

5.1. Testing outliers:

Outliers are data points that significantly differ from the rest of the data, and they can have a substantial impact on the results of statistical analyses and machine learning models. Therefore, it is crucial to identify and handle outliers appropriately. In data analysis, testing for outliers is important to ensure the accuracy and reliability of statistical analyses and machine learning models.

Common techniques for identifying and handling outliers in data analysis include Visualizing the data: Box plots, histograms, and scatter plots can help identify outliers visually. Statistical methods: Z-score, which measures

how many standard deviations a data point is from the mean, and the IQR (Interquartile Range) method, which uses the range between the first and third quartiles, are commonly used statistical techniques to identify outliers. Machine learning techniques: Algorithms such as Isolation Forest and Local Outlier Factor can be used to detect outliers in a dataset. Handling outliers: Once identified, outliers can be handled by removing them if they are due to data entry errors, transforming the data using techniques like winsorization or log transformation, or using robust statistical methods that are less sensitive to outliers.

Data visualization using Boxplot method to identify outliers:

The boxplot is a graphical representation of the distribution of a dataset. It displays the minimum, first quartile (Q1), median, third quartile (Q3), and maximum of a set of data. The boxplot is useful for identifying the central tendency, variability, and skewness of the data, as well as for detecting outliers.

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

# Remove non-numerical and categorical variables
non_numerical_columns = ['Unnamed','id','gender', 'smoke','alco','active','cholesterol','gluc','cardio','BMICat', 'AgeGroup']  # Replace with actual non-
numerical_columns = [col for col in df if col not in non_numerical_columns]

num_plots = len(numerical_columns)
num_cols = 2  # Number of columns for subplots
num_rows = (num_plots + num_cols - 1) // num_cols  # Calculate the number of rows needed

plt.figure(figsize=(15, 10))  # Set the overall figure size
colors = ['skyblue', 'salmon', 'lightgreen', 'lightcoral']  # Define colors for the boxplots
for i, column in enumerate(numerical_columns):
    plt.subplot(num_rows, num_cols, i+1)  # Create subplots
    sns.boxplot(x=df[column], color=colors[i % len(colors)])  # Set color for each boxplot
    plt.title(f'Boxplot for {column}')
plt.tight_layout()  # Adjust the layout
plt.show()
```

Fig. 5. Code for visualization of outliers using box plot

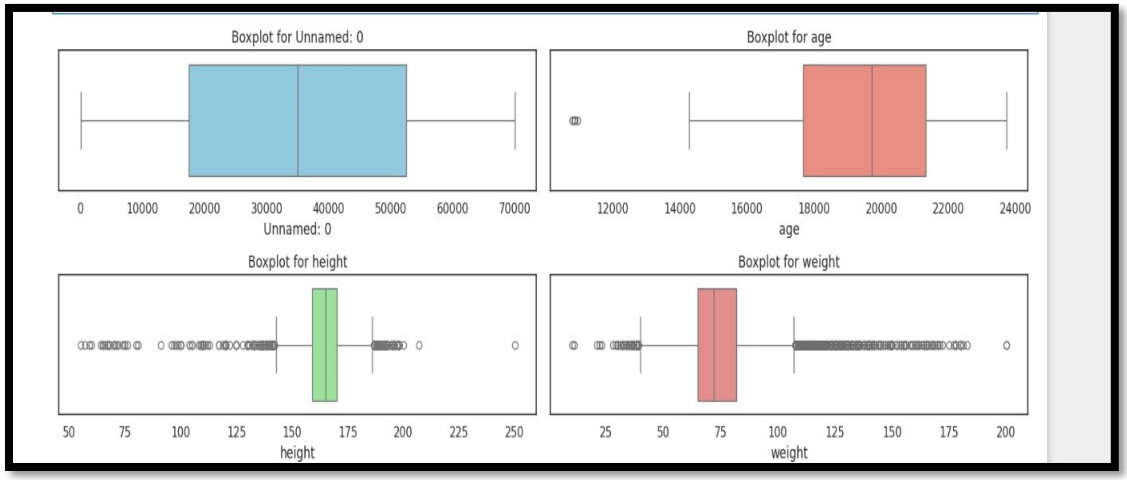

Fig. 6. visualization of outliers using box plot

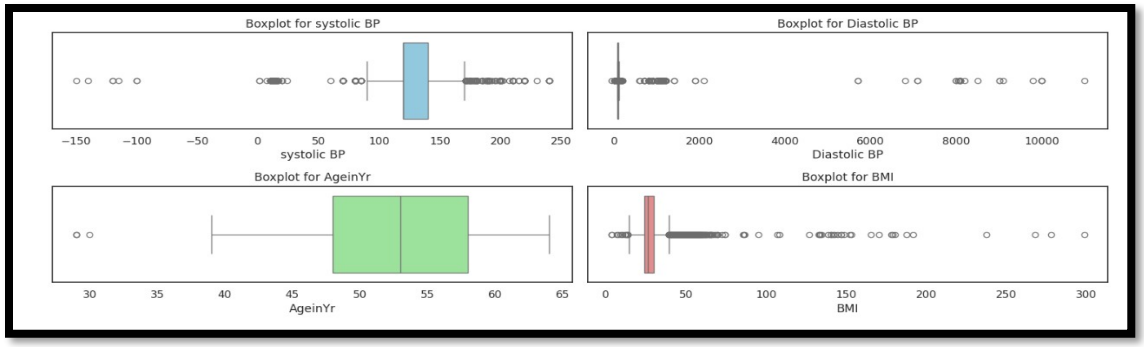

Fig. 7. visualization of outliers using box plot

Visualization of outliers using scattered plot method:

To identify outliers using a scatter plot, we can visually inspect the data for any points that deviate significantly from the overall pattern of the plot. Using the health screening dataset, we have created a scatter plot to visualize the relationship between two variables and identify potential outliers.

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

# Set the aesthetic style of the plots
sns.set(style="white")

# Create scattered plots for numerical variables in subplots
# Remove non-numerical and categorical variables # Replace with actual non-numerical and categorical variables
non_numerical_columns = ['id','gender', 'smoke','alco','active','cholesterol','gluc','cardio','BMICat', 'AgeGroup']
numerical_columns = [col for col in numerical_columns if col not in non_numerical_columns]
num_plots = len(numerical_columns)
num_cols = 3  # Number of columns for subplots
num_rows = (num_plots + num_cols - 1) // num_cols  # Calculate the number of rows needed

plt.figure(figsize=(18, 12))  # Set the overall figure size
for i, column in enumerate(numerical_columns):
    plt.subplot(num_rows, num_cols, i+1)  # Create subplots
# Set color for each scatter plot and adjust transparency and marker size
    sns.scatterplot(x=df.index, y=df[column], alpha=0.7, s=100, color=colors[i % len(colors)])
    #sns.scatterplot(x=data[column], color=colors[i % len(colors)])  # Set color for each boxplot
    plt.title(f'Scatterplot for {column}')
    plt.grid(False)  # Remove grid lines
plt.tight_layout()  # Adjust the layout
plt.show()
```

Fig. 8. Code for visualization of outliers using scatter plot

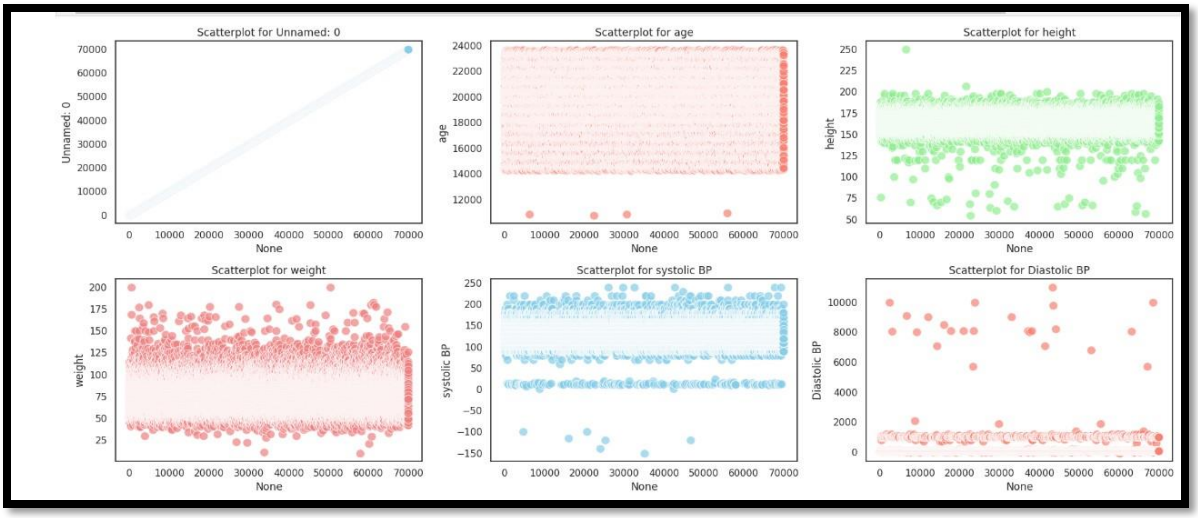

Fig. 9. visualization of outliers using scatter plot

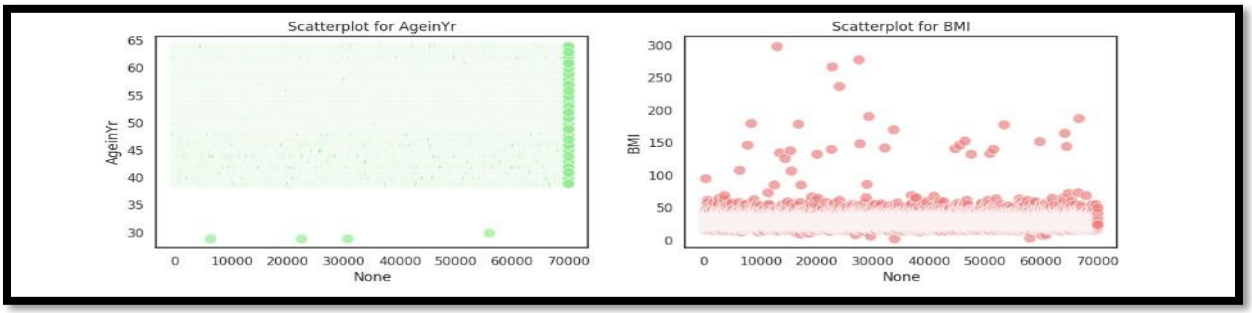

Fig. 9. visualization of outliers using scatter plot

Detecting outliers using Z-score method:

Z-score method measures how many standard deviations a data point is from the mean, and the IQR (Interquartile Range) method, which uses the range between the first and third quartiles, are commonly used statistical techniques to identify outliers.

Identify outliers using the Interquartile Range (IQR) method:

The Interquartile Range (IQR) method is a statistical technique used to identify outliers in a dataset. It involves calculating the IQR, which is the range between the first quartile (Q1) and the third quartile (Q3) of the data. Outliers are then identified as values that fall below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR

```python
# Import necessary libraries
import pandas as pd

# Select numerical columns
numerical_columns = df.select_dtypes(include=['number'])

# Calculate the IQR for numerical columns
Q1 = numerical_columns.quantile(0.25)
Q3 = numerical_columns.quantile(0.75)
IQR = Q3 - Q1

# Identify outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = (numerical_columns< lower_bound) | (numerical_columns> upper_bound)

# Print the outliers for each variable
print(outliers.sum())
```

Fig. 10. code for detecting outliers using Z-score method

```
Unnamed: 0          0
id                  0
age                 4
gender              0
height            519
weight           1819
systolic BP      1395
Diastolic BP     4619
cholesterol         0
gluc            10517
smoke            6169
alco             3762
active          13735
cardio              0
AgeinYr             4
BMI              2033
dtype: int64
```

Fig. 11. Total number of outliers in each variable

Identifying outliers in the health screening dataset using machine learning techniques such as Isolation Forest and Local Outlier Factor:

The output of Isolation Forest and Local Outlier Factor algorithms provides labels for each data point, indicating whether it is an inlier or an outlier. Isolation Forest: The Isolation Forest algorithm assigns a label of 1 to inliers and -1 to outliers. In the provided output: A label of 1 indicates an inlier (data points that are considered typical or normal within the dataset). A label of -1 indicates an outlir (Outliers are data points that significantly deviate from the majority of the data in the dataset).

```python
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor

# Isolation Forest
clf = IsolationForest(contamination=0.1, random_state=42)
outliers_Isolation_Forest = clf.fit_predict(df[['id', 'age','gender', 'height', 'weight', 'systolic BP', 'Diastolic BP','cholesterol', 'gluc', 'smoke', '
print("Outliers identified by Isolation Forest:",outliers_Isolation_Forest)
# Local Outlier Factor
lof = LocalOutlierFactor(n_neighbors=20, contamination=0.1)
outliers_LocalOutlier_Factor = lof.fit_predict(df[['id', 'age','gender', 'height', 'weight', 'systolic BP', 'Diastolic BP','cholesterol', 'gluc', 'smoke'
print("Outliers identified by Local Outlier Factor:",outliers_LocalOutlier_Factor)
```

```
Outliers identified by Isolation Forest: [ 1  1  1 ... -1  1  1]
Outliers identified by Local Outlier Factor: [ 1 -1 -1 ... -1 -1 -1]
```

Fig. 12. Code for identifying outliers using Isolation Forest and Local Outlier Factor and its result

5.1.1 Handling outliers:

We have handled the outliers by imputing them with central tendencies (median), binning, and using robust statistical methods.

```
'''5. Using Robust Statistical Methods'''
#Using median and percentile-based statistics
median = df['AgeinYr'].median()
upper_quartile = df['AgeinYr'].quantile(0.75)
lower_quartile = df['AgeinYr'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['AgeinYr'] > lower_bound) & (df['AgeinYr'] < upper_bound)]

median = df['height'].median()
upper_quartile = df['height'].quantile(0.75)
lower_quartile = df['height'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['height'] > lower_bound) & (df['height'] < upper_bound)]

median = df['weight'].median()
upper_quartile = df['weight'].quantile(0.75)
lower_quartile = df['weight'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['weight'] > lower_bound) & (df['weight'] < upper_bound)]
```

Fig. 13. Code for handling outliers by imputing them with central tendencies

```
median = df['systolic BP'].median()
upper_quartile = df['systolic BP'].quantile(0.75)
lower_quartile = df['systolic BP'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['systolic BP'] > lower_bound) & (df['systolic BP'] < upper_bound)]

median = df['Diastolic BP'].median()
upper_quartile = df['Diastolic BP'].quantile(0.75)
lower_quartile = df['Diastolic BP'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['Diastolic BP'] > lower_bound) & (df['Diastolic BP'] < upper_bound)]

median = df['BMI'].median()
upper_quartile = df['BMI'].quantile(0.75)
lower_quartile = df['BMI'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['BMI'] > lower_bound) & (df['BMI'] < upper_bound)]
```

Fig. 14. Code for handling outliers by imputing them with central tendencies

Outliers' visualization after handling outliers using box plot:

We have minimized the outliers to a larger extent which can be visualized by the below box plot and scatter plots.

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

# Set the aesthetic style of the plots
sns.set(style="white")

# Create scattered plots for numerical variables in subplots
# Remove non-numerical and categorical variables
non_numerical_columns = ['id','gender', 'smoke','alco','active','cholesterol','gluc','cardio','BMICat', 'AgeGroup']  # Replace with actual non-numerical
numerical_columns = [col for col in numerical_columns if col not in non_numerical_columns]
num_plots = len(numerical_columns)
num_cols = 3  # Number of columns for subplots
num_rows = (num_plots + num_cols - 1) // num_cols  # Calculate the number of rows needed

plt.figure(figsize=(18, 12))  # Set the overall figure size
for i, column in enumerate(numerical_columns):
    plt.subplot(num_rows, num_cols, i+1)  # Create subplots
    sns.scatterplot(x=df.index, y=df[column], alpha=0.7, s=100, color=colors[i % len(colors)])  # Set color for each scatter plot and adjust transparency
    #sns.scatterplot(x=data[column], color=colors[i % len(colors)])  # Set color for each boxplot
    plt.title(f'Scatterplot for {column}')
    plt.grid(False)  # Remove grid lines
plt.tight_layout()  # Adjust the layout
plt.show()
```

Fig. 15. Code for Outliers' visualization after handling outliers using box plot

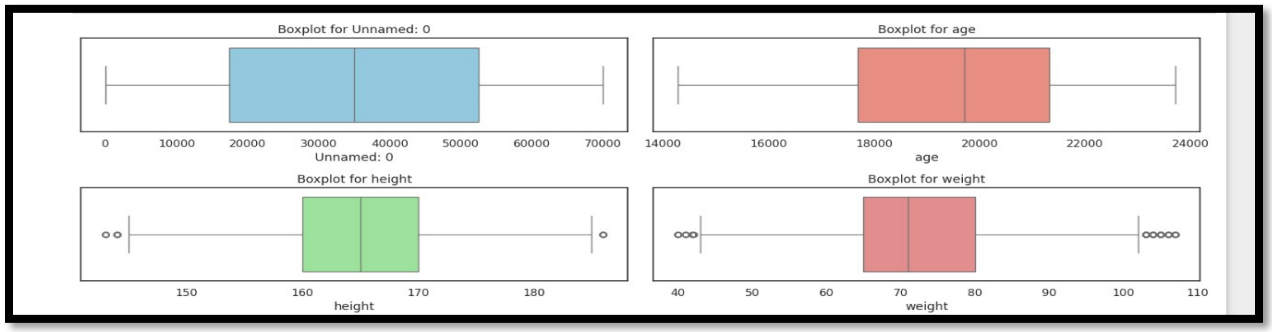

Fig. 16. Outliers' visualization after handling outliers using box plot

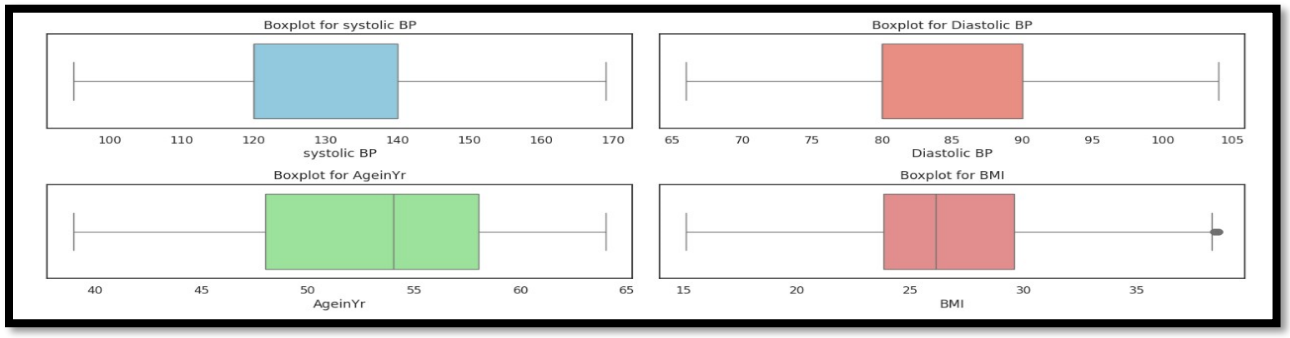

Fig. 17. Outliers' visualization after handling outliers using box plot

Outliers visualization after handling using scatter plot:

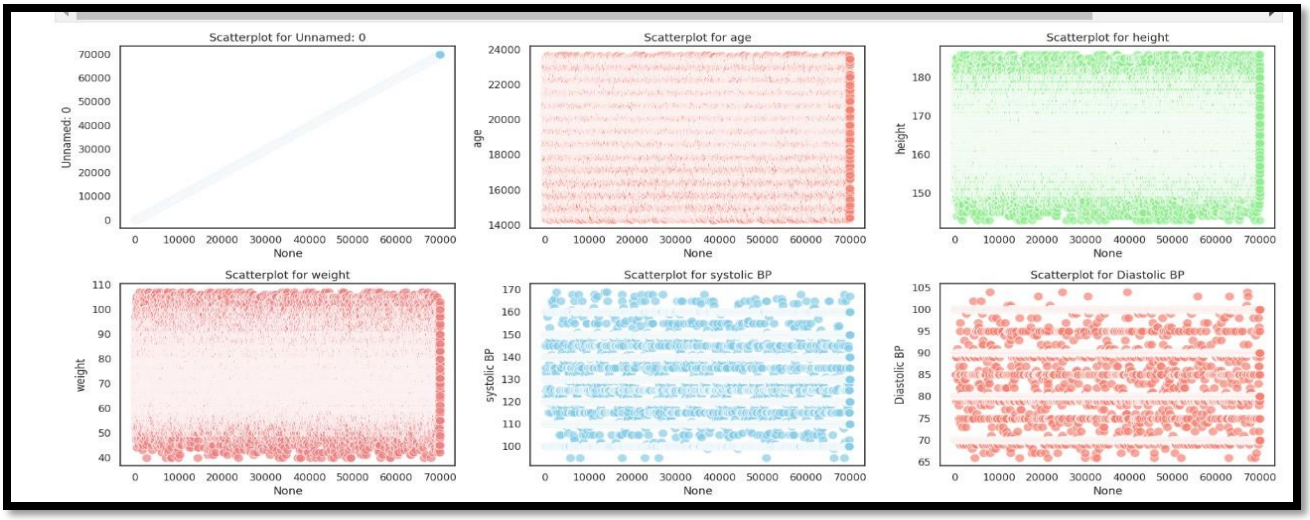

Fig. 18. Outliers' visualization after handling outliers using scatter plot

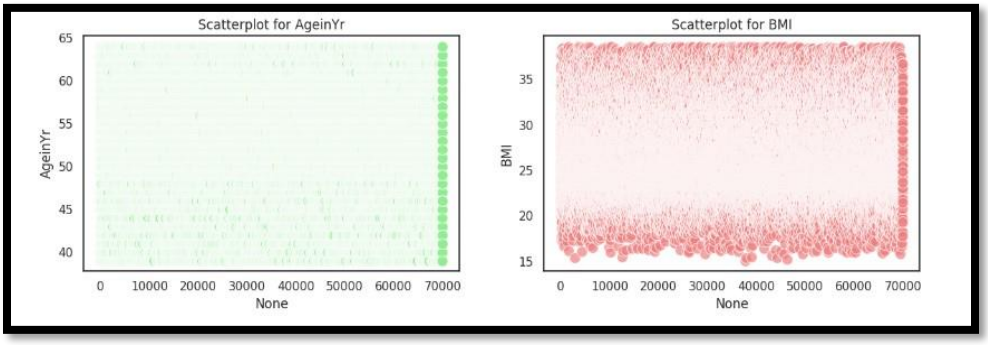

Fig. 19. Outliers' visualization after handling outliers using scatter plot

5.2. Summary statistics of the data after handling the outliers:

```
'''describe.T will transpose the describe function result'''
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	60645.0	34983.674631	20217.237035	0.0	17411.0	35001.0	52487.0	69999.0
id	60645.0	49949.836639	28865.401280	0.0	24876.0	50004.0	74869.0	99999.0
age	60645.0	19480.570731	2458.382527	14282.0	17696.0	19710.0	21329.0	23713.0
gender	60645.0	1.354737	0.478437	1.0	1.0	1.0	2.0	2.0
height	60645.0	164.581070	7.474070	143.0	160.0	165.0	170.0	186.0
weight	60645.0	72.694129	11.829999	40.0	65.0	71.0	80.0	107.0
systolic BP	60645.0	126.046220	13.650576	95.0	120.0	120.0	140.0	169.0
Diastolic BP	60645.0	81.585753	7.541908	66.0	80.0	80.0	90.0	104.0
cholesterol	60645.0	1.350482	0.669660	1.0	1.0	1.0	1.0	3.0
gluc	60645.0	1.217578	0.564985	1.0	1.0	1.0	1.0	3.0
smoke	60645.0	0.087542	0.282630	0.0	0.0	0.0	0.0	1.0
alco	60645.0	0.052189	0.222410	0.0	0.0	0.0	0.0	1.0
active	60645.0	0.804667	0.396460	0.0	1.0	1.0	1.0	1.0
cardio	60645.0	0.488284	0.499867	0.0	0.0	0.0	1.0	1.0
AgeinYr	60645.0	52.872883	6.742409	39.0	48.0	54.0	58.0	64.0
BMI	60645.0	26.857128	4.199078	15.1	23.8	26.1	29.6	38.6

Fig. 20. Summary statistics of the data

- The mean age of the individuals in the dataset is approximately 52.87 years, with a standard deviation of 6.74 years. The age ranges from 39 to 64 years.
- The mean BMI (Body Mass Index) is around 26.86, with a standard deviation of 4.20. The BMI values range from 15.1 to 38.6.
- The mean systolic blood pressure is approximately 126.05 mmHg, with a standard deviation of 13.65 mmHg. The values range from 95.0 to 169.0 mmHg.
- The mean diastolic blood pressure is about 81.59 mmHg, with a standard deviation of 7.54 mmHg. The values range from 66.0 to 104.0 mmHg.
- The prevalence of smoking and alcohol consumption is relatively low in the dataset, with mean values of 0.088 and 0.052, respectively.
- The mean cholesterol level is approximately 1.35, and the mean glucose level is around 1.22.
- The mean height is 164.58 cm, and the mean weight is 72.69 kg.

5.3. Testing Normality of the Data

The normality test in data analysis is a statistical method used to determine if a given dataset follows a normal distribution. A normal distribution, also known as a Gaussian distribution, is characterized by a symmetric, bell-shaped curve. Many statistical methods and models assume that the data is normally distributed, making the normality test a major step in data analysis.

Kurtosis and Skewness: Kurtosis measures the tailedness of the distribution, while skewness measures the asymmetry of the distribution. For a normal distribution, the kurtosis should be close to 0, and the skewness should be close to 0.

a. Shapiro-Wilk Test:

One commonly used normality test is the Shapiro-Wilk test, which assesses whether a sample comes from a normally distributed population. The test provides a p-value, and if the p-value is less than a chosen significance level (e.g., 0.05), the null hypothesis that the data is normally distributed is rejected.

We imported the “Health screening Data.csv” into Jupyter Python Notebook and imported “Pandas” as pd and scipy.stats form Shapiro. Stored the data in data frame column, converted all non-numeric data to numeric data, and checked for data points. If the data points were found to be less than three, the message was printed. If there are enough data points, then the data is flattened (to make it suitable for analysis). After flattening the data, Shapiro-wilk test is performed and the results are printed. After printing the results, the p value is checked to detect whether the data is normally distributed or not normally distributed. Code is included in the screenshot below:

```
#Shapiro-Wilk test:
from scipy import stats
import numpy as np
from scipy.stats import shapiro
#Convert all columns to numeric (ignoring non-numeric values)
df = df.apply(pd.to_numeric, errors='coerce')

# Drop columns with less than 3 non-NaN values
df = df.dropna(thresh=3, axis=1)

# Check if there are enough data points
if len(df) < 3:
    print("Insufficient data points for normality test.")
else:
    # Flatten the DataFrame into a one-dimensional array
    data = df.values.flatten()

    # Perform Shapiro-Wilk test
    statistic, p_value = shapiro(data)

    # Print the results
    print(f'Statistic: {statistic}, p-value: {p_value}')

    # Check the p-value against a significance level (e.g., 0.05)
    if p_value > 0.05:
        print("The dataset appears to be normally distributed.")
    else:
        print("The dataset does not appear to be normally distributed.")
```

```
Statistic: 0.4478346109390259, p-value: 0.0
The dataset does not appear to be normally distributed.
```

Fig. 21. Code for Shapiro-Wilk Test and its result

b. Visual Inspection of Normality:

Plotting histograms, Q-Q plots, or density plots of the data can provide a visual indication of whether the data is approximately normally distributed. A bell-shaped curve in the histogram or a straight line in the Q-Q plot suggests normality.

Histogram -

Plot the histogram for the column with 20 bins and present it. The histogram was plotted to check the normality and distribution of the data set. Code is given in the screenshot below. Plotting histograms, Q-Q plots, or density plots of the data can provide a visual indication of whether the data is approximately normally distributed. A bell-shaped curve in the histogram suggests normality. While histograms can provide an initial indication of normality, they have limitations, especially with small sample sizes.

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, roc_auc_score

# Select continuous numerical variables
numerical_vars = ['age', 'height', 'weight', 'systolic BP', 'Diastolic BP', 'AgeinYr', 'BMI']

# Create subplots for each variable
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 10))

# Plot histograms for each variable
for i, var in enumerate(numerical_vars):
    row = i // 3
    col = i % 3
    axes[row, col].hist(df[var], bins=10, alpha=0.7, density=True, color='b', edgecolor='black', label='Actual Distribution')
    axes[row, col].set_title(var)

    # Add normal distribution line
    mu, sigma = df[var].mean(), df[var].std()
    xmin, xmax = df[var].min(), df[var].max()
    x = np.linspace(xmin, xmax, 100)
    p = stats.norm.pdf(x, mu, sigma)
    axes[row, col].plot(x, p, 'k', linewidth=2, label='Normal Distribution')
    axes[row, col].legend()

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

Fig. 22. Code for Histogram plotting

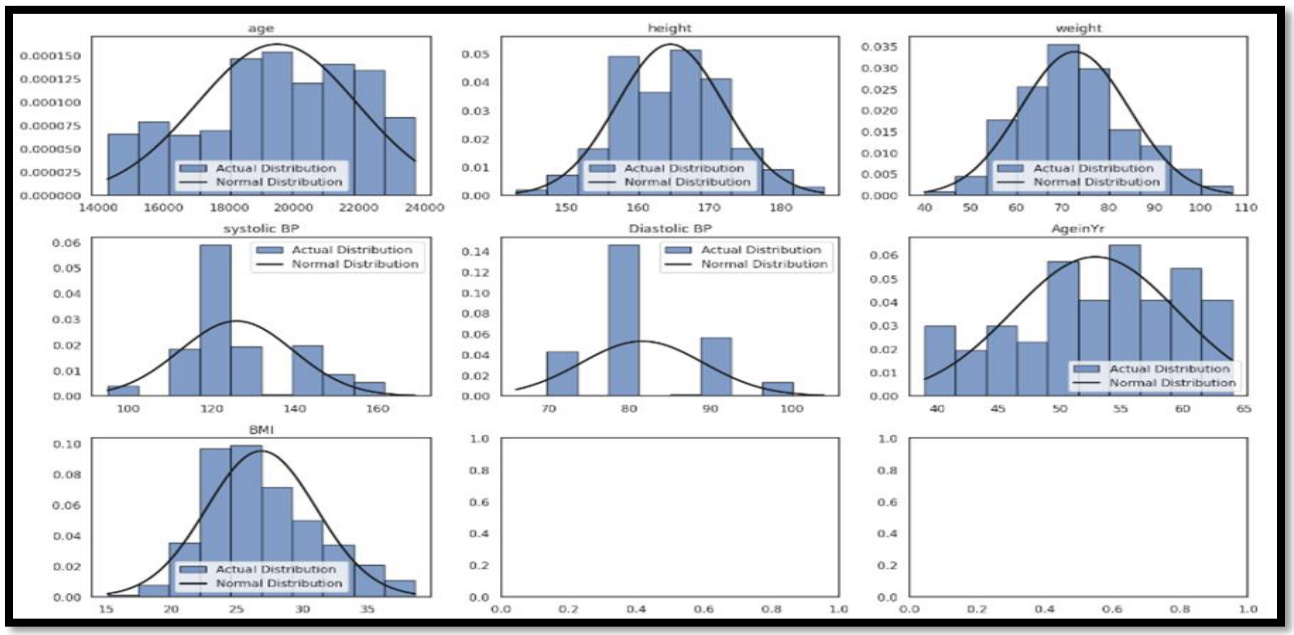

Fig. 23. Histogram of the variables

Q-Q Plot -

Normal probability plots, also known as Q-Q (quantile-quantile) plots, are a better choice for assessing normality. A straight line in the Q-Q plot suggests normal distribution of the data.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm


# Select numerical columns excluding specific categorical variables
numerical_data = df.select_dtypes(include=['number']).drop(columns=['id', 'gender', 'smoke', 'alco', 'active', 'cholesterol', 'gluc', 'cardi

# Plot QQ-Plots for selected numerical variables
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 15))
for i, column in enumerate(numerical_data.columns):
    ax = axes[i // 3, i % 3]
    sm.qqplot(numerical_data[column], line ='45', ax=ax)
    ax.set_title(column)
plt.tight_layout()
plt.show()
```

Fig. 24. Code for Q-Q plot plotting

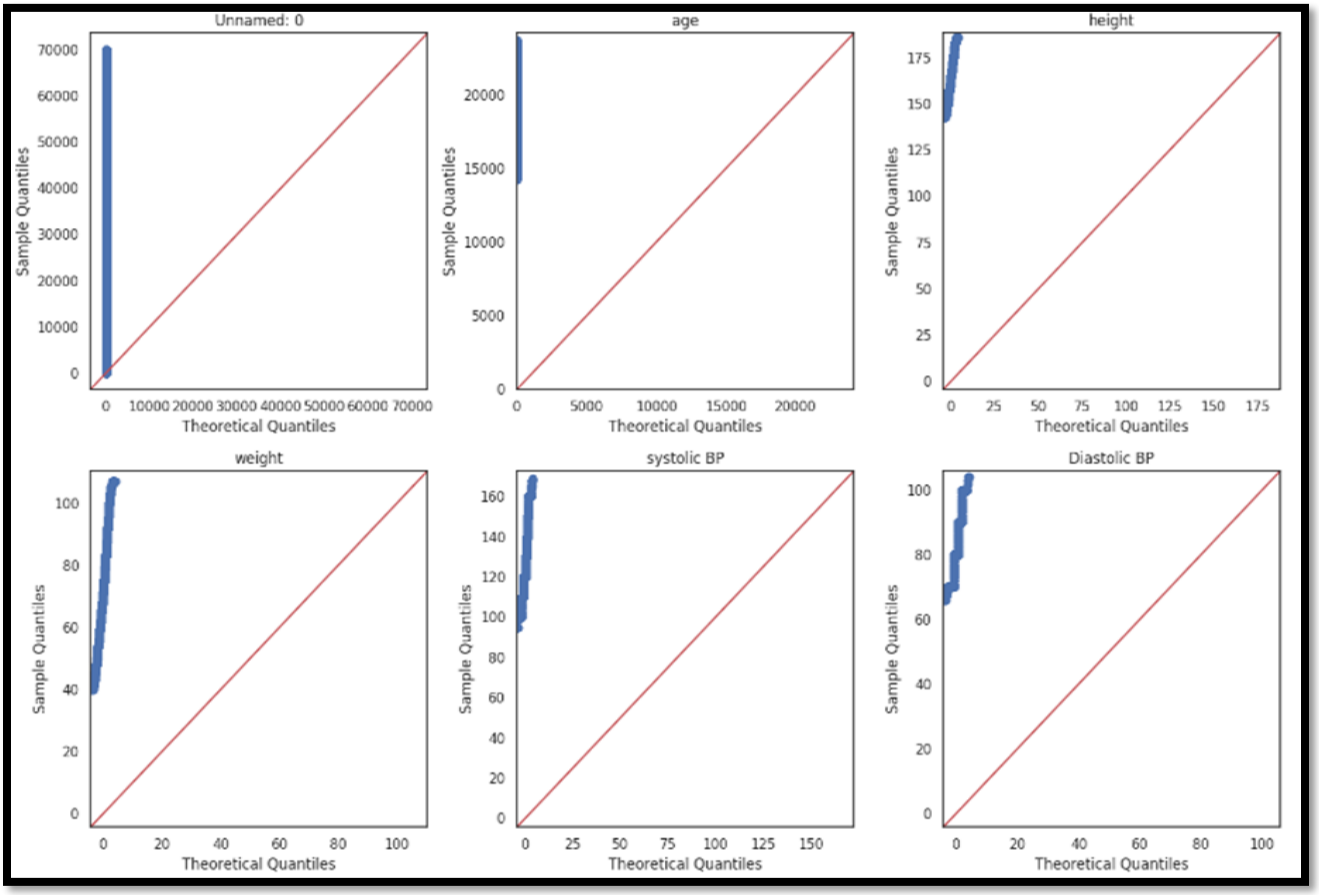

Fig. 24. Q-Q normality plots for different variables

c. Testing Kurtosis and Skewness of the dataset:

Skewness measures the asymmetry of the distribution. A skewness value of 0 indicates a perfectly symmetrical distribution. A positive skewness indicates that the distribution is skewed to the right, and a negative skewness indicates that the distribution is skewed to the left.

Kurtosis is a measure of the "heaviness" of the tails of a distribution. A kurtosis value of 3 indicates a normal distribution. Excess kurtosis (kurtosis min3) greater than 0 indicates heavier tails than a normal distribution, and less than 0 indicates lighter tails.

```
import pandas as pd
from scipy.stats import kurtosis, skew

# Calculate the kurtosis of the dataset
data_kurtosis = df.kurtosis()
print('kurtosis of the data:')
print(data_kurtosis)

# Calculate the skewness of the dataset
data_skewness = df.skew()
print('skewness of the data:')
print(data_skewness)
```

Fig. 25. Code to test Kurtosis and Skewness of the dataset

```
kurtosis of the data:
Unnamed: 0          -1.201599
id                  -1.199981
age                 -0.819325
gender              -1.631288
height              -0.203171
weight              -0.171220
systolic BP          0.041878
Diastolic BP         0.103441
cholesterol          1.236241
gluc                 4.650124
smoke                6.519633
alco                14.217471
active               0.362329
cardio              -1.997868
AgeinYr             -0.814585
BMI                 -0.191593
dtype: float64
```

```
skewness of the data:
Unnamed: 0          0.000614
id                 -0.000659
age                -0.309389
gender              0.607261
height              0.061609
weight              0.353174
systolic BP         0.698882
Diastolic BP        0.454442
cholesterol         1.659393
gluc                2.472117
smoke               2.918804
alco                4.027034
active             -1.536983
cardio              0.046877
AgeinYr            -0.308066
BMI                 0.524557
dtype: float64
```

Fig. 26 and 27. Result of Kurtosis and Skewness of the dataset

Based on the kurtosis values, the variables "cholesterol", "gluc", "smoke", "alco" have kurtosis values greater than 3, indicating that they have heavy tails and are leptokurtic. This suggests that these variables have more extreme values or outliers compared to a normal distribution.

Based on the skewness values, the variables "gender", "weight", "systolic BP", "Diastolic BP", "cholesterol", "gluc", "smoke", "alco", "BMI" have skewness values greater than 1, indicating that they are highly skewed. This suggests that these variables have a non-normal distribution and are skewed to the right.

d. Kolmogorov-Smirnov Test to test normality:

The Kolmogorov-Smirnov test is used to determine whether a dataset follows a specific probability distribution or if two datasets have the same distribution. It is a non-parametric test that compares the cumulative distribution function (CDF) of the data with the CDF of the expected distribution.

The ks_statistic variable will contain the Kolmogorov-Smirnov test statistic, and the p_value variable will contain the p-value for the test.

KS Statistic: The KS statistic measures the maximum difference between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution. A larger KS statistic indicates a greater difference between the sample and the reference distribution.

P-Value: The p-value indicates the probability of observing the KS statistic or a more extreme value if the sample were drawn from the reference distribution. A small p-value suggests that the sample distribution significantly differs from the reference distribution

When conducting a KS test, if the p-value is less than the chosen significance level (e.g., 0.05), there is evidence to reject the null hypothesis that the sample follows the reference distribution. Conversely, if the p-value is greater than the significance level, there is no significant evidence to reject the null hypothesis.

In the context of the provided dataset "Health Screening Data.csv," the KS test was used to assess whether a particular variable follows the normal distribution. The KS statistic and p-value helped to determine the goodness of fit of the data to the reference distribution. As the P-value was less that 0.05, it was identified that the data was not normally distributed.

```
from scipy.stats import kstest

# Perform the Kolmogorov-Smirnov test for a specific column (e.g., 'age')
ks_statistic, p_value = kstest(df['BMI'], 'norm')
print("KS Statistic:", ks_statistic)
print("P-Value:", p_value)
# Check the p-value against a significance level (e.g., 0.05)
if p_value > 0.05:
    print("The dataset appears to be normally distributed.")
else:
    print("The dataset does not appear to be normally distributed.")
```

```
KS Statistic: 1.0
P-Value: 0.0
The dataset does not appear to be normally distributed.

Descriptive Statistics
```

Fig. 28. Code for Kolmogorov-Smirnov Test and result

5.4. Relationship analysis between variables:

a. Correlation matrix:

In order to check the correlation among the attributes, we created correlation matrix for the numerical data and we plotted the heatmap for the dataset “Health Screening Data.CSV”. We found that the risk factors (age, gender, BMI, ap_hi (systolic BP), ap_lo (Diastolic_BP), cholesterol, gluc (glucose), smoke (smoking), alco (alcohol) and active (physical activity)).

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
df = pd.read_csv('Health Screening Data.csv')
numerical_data = df.select_dtypes(include=['int64', 'float64'])
correlation_matrix = numerical_data.corr()
plt.figure(figsize=(10, 8))
heatmap = sns.heatmap(correlation_matrix, annot=True, square = True, fmt = '.2f', annot_kws={'size':10},cmap="coolwarm")
plt.title('Correlation between risk factors and cardiovascular disease')
plt.show()
print (correlation_matrix)
```

Fig. 29. Code for correlation matrix

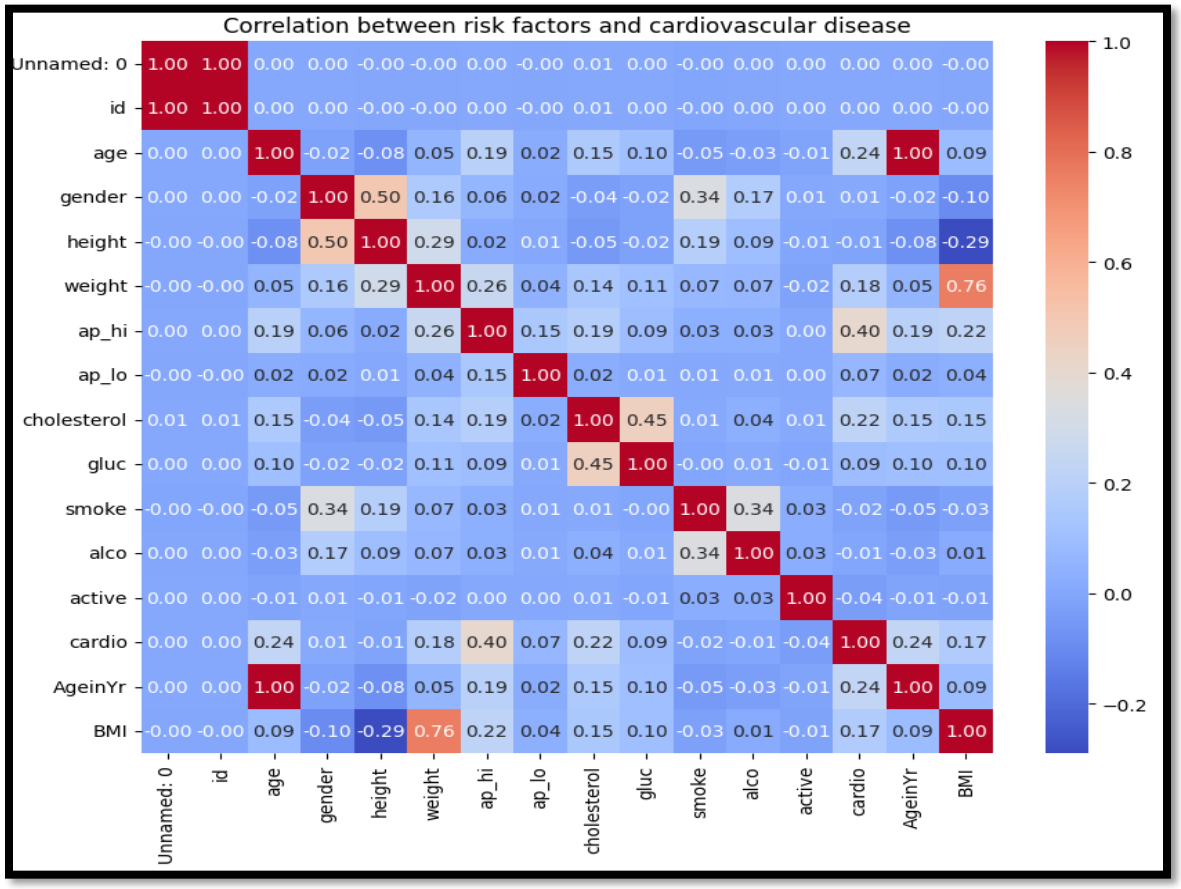

Fig. 30. correlation matrix heat map

From the correlation heat map, we have identified that the attributes of ap_hi (systolic blood pressure), Age, cholesterol, weight, BMI have strong correlation with cardio (cardiovascular disease) and the attributes of gluc (glucose), smoke, alco (alcohol), active (physican activity) has weak correlation with cardio (cardiovascular disease) and the attributes of ID has no correlation with cardio (cardiovascular disease).

b. Kendall's Rank and Spearman's Correlation tests:

The Kendall (1955) rank correlation coefficient evaluates the degree of similarity between two sets of ranks given to a same set of objects. Spearman's rank correlation measures the strength and direction of association between two ranked variables.

To identify the strength and direction of monotonic association between independent and dependent variables, and to determine the correlation coefficient, we performed Kendall's Rank and Spearman's Correlation tests.

Both Kendall's tau and Spearman's rho indicate a moderate positive rank correlation between the variables being tested, and the low p-values suggest that these correlations are statistically significant.

```python
import pandas as pd
from scipy.stats import kendalltau
import matplotlib.pyplot as plt
import seaborn as sns

# Select the independent variables and the dependent variable
independent_vars = ['height', 'weight', 'cholesterol','systolic BP', 'Diastolic BP', 'AgeinYr', 'BMI','smoke', 'alco','active']
dependent_var = 'cardio'  # Assuming 'cardio' is the categorical dependent variable

# Calculate Kendall's tau coefficient for each independent variable
correlations = {}
for var in independent_vars:
    tau, p_value = kendalltau(df[var], df[dependent_var])
    correlations[var] = tau

# Visualize the result
plt.figure()
plt.bar(correlations.keys(), correlations.values())
plt.xlabel('Independent Variable')
plt.ylabel("Kendall's Tau Coefficient")
plt.title("Kendall's Tau Coefficient for Each Independent Variable")
plt.xticks(rotation=45)  # Rotate the x-axis labels for clarity
plt.show()

# Print the result
print("Kendall's rank correlation coefficient:", tau)
print("P-value:", p_value)
```

Fig. 30. Code for Kendall's Rank correlation

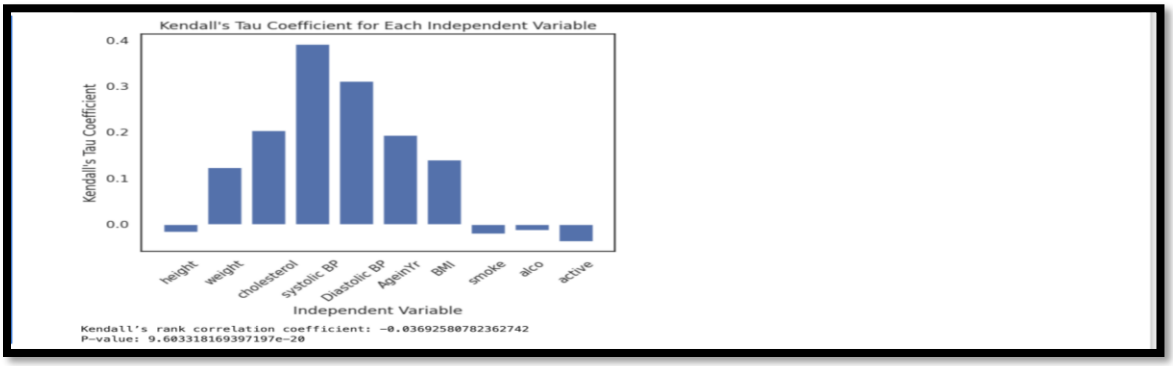

Fig. 31. Barplot showing Kendall's Tau Coefficient between variables

```python
import pandas as pd
from scipy.stats import spearmanr
import seaborn as sns
import matplotlib.pyplot as plt


# Select the independent continuous variables and the dichotomous dependent variable
independent_vars = ['height', 'weight', 'cholesterol','systolic BP', 'Diastolic BP', 'AgeinYr', 'BMI','smoke', 'alco','active']
dependent_var = 'cardio'  # Assuming 'cardio_disease' is the dichotomous dependent variable

# Calculate Spearman's rank correlation for each independent variable
correlations = {}
for var in independent_vars:
    rho, p_value = spearmanr(df[var], df[dependent_var])
    correlations[var] = rho

# Visualize the correlations
plt.figure()
plt.bar(correlations.keys(), correlations.values())
plt.xlabel('Independent Variable')
plt.ylabel("Spearman's Rank Correlation Coefficient")
plt.title("Spearman's Rank Correlation for Each Independent Variable")
plt.xticks(rotation=45)  # Rotate the x-axis labels for clarity
plt.show()

# Print the result
print("Spearman's Rank Correlation Coefficient:", rho)
print("P-value:", p_value)
```

Fig. 32. Code for Spearman correlation test

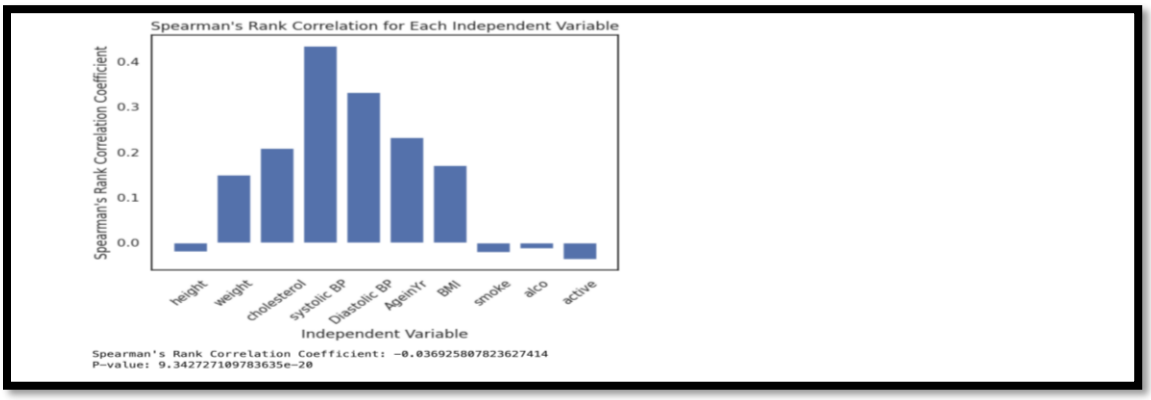

Fig. 33. Barplot showing Spearman's Rank Correlation between variables

This suggests a very weak, but statistically significant, negative relationship between variables analyzed. While the correlation is significant, the strength of the relationship is very low, indicating that the variables are not strongly related in terms of their ranks.

5.5. Hypothesis testing:

Null Hypothesis (H0): There is no significant relationship between the health parameters (age, gender, blood pressure, BMI, cholesterol level, smoking, alcohol consumption, physical activity etc.) and the risk of cardiovascular disease (CVD) in the study population.

Alternative Hypothesis (H1): There is a significant relationship between the health parameters (age, gender, blood pressure, BMI, cholesterol level, smoking, alcohol consumption, physical activity etc.) and the risk of cardiovascular disease (CVD) in the study population.

As the data is not normally distributed, we have performed non-parametric tests: Maan Whitney U test (Wilcoxon Rank Sum test), Kruskal Walls test, Wilcoxon Signed-Rank test, Friedman test and Chi-square test Robust regression Boot strapping.

Mann - Whitney U Test (Wilcoxon Rank Sum test), Kruskal Walls test, Friedman test and Wilcoxon Signed-Rank test is used to compare differences between two independent groups when the dependent variable is either ordinal or continuous, but not normally distributed. In our dataset, since the dependent variable has categorical values, these tests were not appropriate for hypothesis testing.

a. Chi-Square test:

It is a non-parametric test that determines whether there is an association between categorical variables. In the analysis of health screening dataset, the chi-square test was used to analyze the relationship between various health indicators and the presence of cardiovascular disease. This test identified the significant relationship between physical activity, smoking, glucose levels, cholesterol levels, alcoholism and cardiovascular disease.

```
import pandas as pd
from scipy.stats import chi2_contingency


# chi-square test on the 'smoke' and 'cardio' variables
contingency_table = pd.crosstab(df['smoke'], df['cardio'])

# Perform the chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Statistic:", chi2)
print("P-Value:", p)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
if p > 0.05:
    print('there is no significant relationship between smoking and cardiovascular diseasee')
else:
    print('there is Significant relationship between smoking and cardiovascular disease')
```

```
Chi-square Statistic: 26.70904257194477
P-Value: 2.3651209457241988e-07
Degrees of Freedom: 1
Expected Frequencies: [[28316.30122846 27019.69877154]
 [ 2716.69877154  2592.30122846]]
there is Significant relationship between smoking and cardiovascular disease
```

Fig. 34. Code for Chi-square test between smoking and cardiovascular disease and its result

```
import pandas as pd
from scipy.stats import chi2_contingency


# chi-square test on the 'smoke' and 'cardio' variables
contingency_table = pd.crosstab(df['cholesterol'], df['cardio'])

# Perform the chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Statistic:", chi2)
print("P-Value:", p)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
if p > 0.05:
    print('there is no significant relationship between cholesterol levels and cardiovascular disease')
else:
    print('there is a significant relationship between cholesterol levels and cardiovascular disease')
```

```
Chi-square Statistic: 2840.2199453085404
P-Value: 0.0
Degrees of Freedom: 2
Expected Frequencies: [[23582.41907824 22502.58092176]
 [ 4024.64415863  3840.35584137]
 [ 3425.93676313  3269.06323687]]
there is a significant relationship between cholesterol levels and cardiovascular disease
```

Fig. 34. Code for Chi-square test between cholesterol levels and cardiovascular disease and its result

```
import pandas as pd
from scipy.stats import chi2_contingency


# chi-square test on the 'smoke' and 'cardio' variables
contingency_table = pd.crosstab(df['gluc'], df['cardio'])

# Perform the chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Statistic:", chi2)
print("P-Value:", p)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
if p > 0.05:
    print('there is no significant relationship between glucose levels and cardiovascular disease')
else:
    print('there is a significant relationship between glucose levels and cardiovascular disease')
```

```
Chi-square Statistic: 453.8814583807049
P-Value: 2.7599015700232877e-99
Degrees of Freedom: 2
Expected Frequencies: [[26592.33095886 25374.66904114]
 [ 2129.24912194  2031.75087806]
 [ 2311.4199192   2205.5800808 ]]
there is a significant relationship between glucose levels and cardiovascular disease
```

Fig. 35. Code for Chi-square test between glucose and cardiovascular disease and its result

```
import pandas as pd
from scipy.stats import chi2_contingency


# chi-square test on the 'smoke' and 'cardio' variables
contingency_table = pd.crosstab(df['alco'], df['cardio'])

# Perform the chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Statistic:", chi2)
print("P-Value:", p)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
if p > 0.05:
    print('there is no significant relationship between alcoholism and cardiovascular disease')
else:
    print('there is a significant relationship between alcoholism and cardiovascular disease')
```

```
Chi-square Statistic: 10.312780967080396
P-Value: 0.0013211205168062103
Degrees of Freedom: 1
Expected Frequencies: [[29413.41973782 28066.58026218]
 [ 1619.58026218  1545.41973782]]
there is a significant relationship between alcoholism and cardiovascular disease
```

Fig. 36. Code for Chi-square test between alcoholism and cardiovascular disease and its result

```
import pandas as pd
from scipy.stats import chi2_contingency


# chi-square test on the 'smoke' and 'cardio' variables
contingency_table = pd.crosstab(df['active'], df['cardio'])

# Perform the chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Statistic:", chi2)
print("P-Value:", p)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
if p > 0.05:
    print('there is no significant relationship between physical activity and cardiovascular disease')
else:
    print('there is a significant relationship between physical activity and cardiovascular disease')
```

```
Chi-square Statistic: 82.50415926756672
P-Value: 1.0544801043633886e-19
Degrees of Freedom: 1
Expected Frequencies: [[ 6061.78445049  5784.21554951]
 [24971.21554951 23827.78445049]]
there is a significant relationship between physical activity and cardiovascular disease
```

Fig. 37. Code for Chi-square test between physical activity and cardiovascular disease and its result

Kruskal-Wallis H test:

Kruskal-Wallis H test (sometimes also called the "one-way ANOVA on ranks") is a rank-based nonparametric test that can be used to determine if there are statistically significant differences between two or more groups of an independent variable on a continuous or ordinal dependent variable. As our dependent variable as categorical values, this test was not performed on our dataset.

Wilcoxon Signed-Rank test:

This non-parametric test is used to compare the distributions of two related groups when the dependent variable is continuous, and the independent variable is dichotomous. Therefore, it was not performed for our dataset.

Logistic regression model:

Logistic regression is used to predict categorical dependent variables. It is used when the prediction is categorical, for example, yes or no, true, or false, 0 or 1. logistic regression is used for categorical outcome variables, such as

death. Independent variables can be continuous, categorical, or a mix of both. Logistic regression model was used for our dataset as nature of the dependent variable (cardio) is categorical and binary.

```
import pandas as pd
import statsmodels.api as sm


# Select the independent variables (health parameters) and the dependent variable (CVD status)
independent_vars = ['AgeinYr', 'gender', 'height', 'weight', 'systolic BP', 'Diastolic BP', 'cholesterol', 'gluc', 'smoke', 'alco', 'active']
dependent_var = 'cardio'

# Add a constant term to the independent variables
independent_vars = sm.add_constant(df[independent_vars])

# Fit the logistic regression model
logit_model = sm.Logit(df[dependent_var], independent_vars)
logit_result = logit_model.fit()

# Print the summary of the logistic regression model
print(logit_result.summary())

if p > 0.05:
    print('there is no significant relationship between health indicators and cardiovascular disease')
else:
    print('there is a significant relationship between health indicators and cardiovascular disease')
```

Fig. 38. Code for logistic regression model

```
Optimization terminated successfully.
         Current function value: 0.563739
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                 cardio   No. Observations:                60645
Model:                          Logit   Df Residuals:                    60633
Method:                           MLE   Df Model:                           11
Date:                Sat, 02 Dec 2023   Pseudo R-squ.:                 0.1864
Time:                        17:03:33   Log-Likelihood:                -34188.
converged:                       True   LL-Null:                       -42019.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -12.3733      0.276    -44.887      0.000     -12.914     -11.833
AgeinYr        0.0515      0.001     35.776      0.000       0.049       0.054
gender        -0.0223      0.023     -0.950      0.342      -0.068       0.024
height        -0.0049      0.002     -3.156      0.002      -0.008      -0.002
weight         0.0123      0.001     13.998      0.000       0.011       0.014
systolic BP    0.0642      0.001     59.108      0.000       0.062       0.066
Diastolic BP   0.0142      0.002      7.894      0.000       0.011       0.018
cholesterol    0.5002      0.017     29.851      0.000       0.467       0.533
gluc          -0.1155      0.019     -6.098      0.000      -0.153      -0.078
smoke         -0.1675      0.037     -4.513      0.000      -0.240      -0.095
alco          -0.2250      0.045     -4.953      0.000      -0.314      -0.136
active        -0.2337      0.023    -10.046      0.000      -0.279      -0.188
==============================================================================
there is a significant relationship between health indicators and cardiovascular disease
```

Fig. 39. Results of the logistic regression model

The provided logistic regression results show that included health indicators are significantly associated with the presence of cardiovascular disease. The direction and magnitude of the coefficients provided information about the strength and nature of these relationships, and the model fit suggests that the included variables explain a moderate amount of the variance in the presence of cardiovascular disease.

Significant Variables:

The coefficients of the variables "AgeinYr", "height", "weight", "systolic BP", "Diastolic BP", "cholesterol", "gluc", "smoke", "alco", and "active" are all statistically significant ($P < 0.05$). This suggests that these variables are associated with the presence of cardiovascular disease.

Effect Direction:

The positive coefficients (e.g., "systolic BP", "Diastolic BP", "cholesterol") indicate that an increase in these variables is associated with a higher likelihood of cardiovascular disease, while the negative coefficients (e.g., "gluc", "smoke", "alco", "active") suggest a lower likelihood of cardiovascular disease with an increase in these variables.

Magnitude of Effect:

The magnitude of the coefficients provides information about the strength of the relationship between the independent variables and the presence of cardiovascular disease. For example, "systolic BP" has a relatively large coefficient (0.0642), indicating a stronger effect compared to other variables.

Model Fit:

The Pseudo R-squared value of 0.1864 suggests that the model explains approximately 18.64% of the variance in the presence of cardiovascular disease. It indicates that there are other factors not accounted for in the model.

6. Data visualization

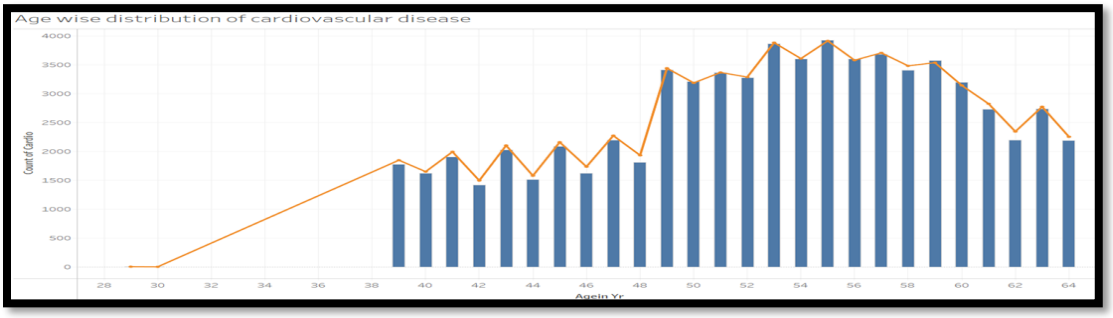

Fig. 40. Age wise distribution of cardiovascular disease

The age group of 50 to 60 years showed high cardiovascular disease in the dataset.

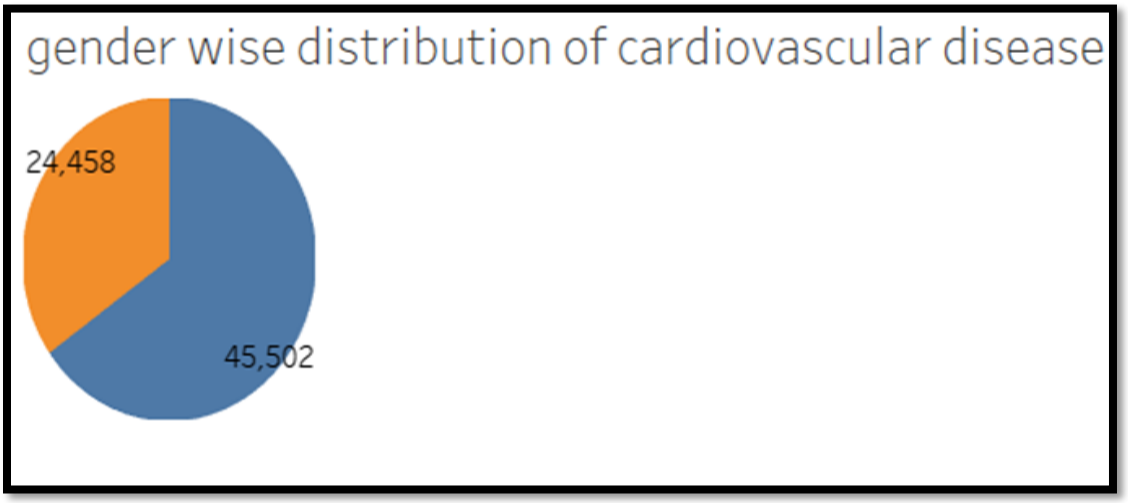

Fig. 41. Gender wise distribution of cardiovascular disease

Males have high incidence of cardiovascular disease compared to females in the dataset.

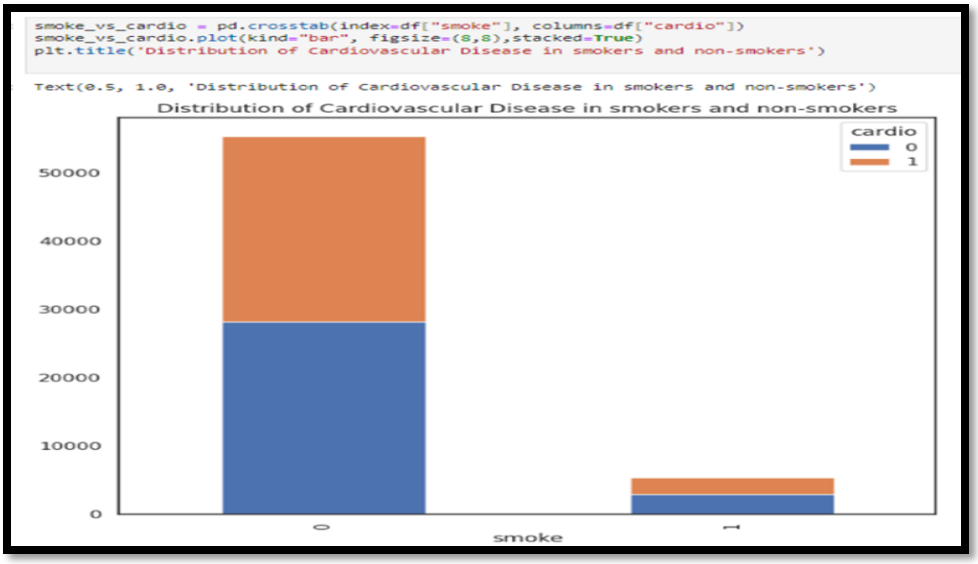

Fig. 42. Distribution of cardiovascular disease in smokers and non-smokers

The above figure shows there is high distribution of cardiovascular disease in non-smokers compared to smokers.

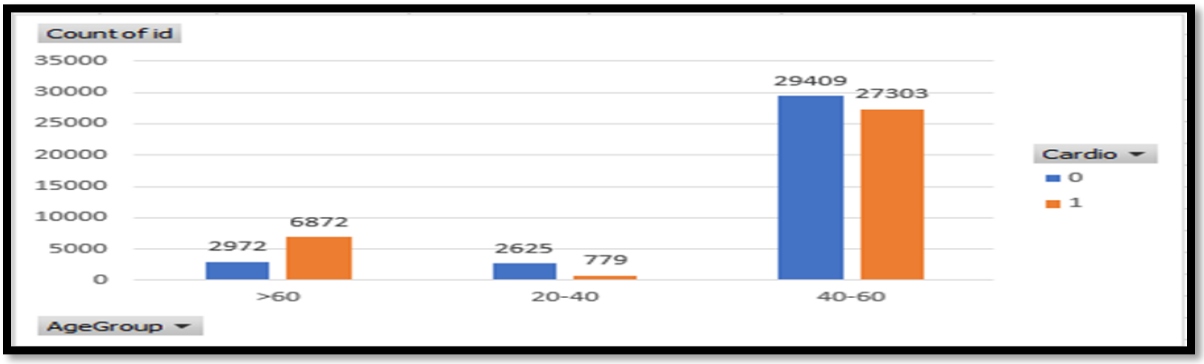

Fig. 43. Distribution of cardiovascular disease in smokers and non-smokers

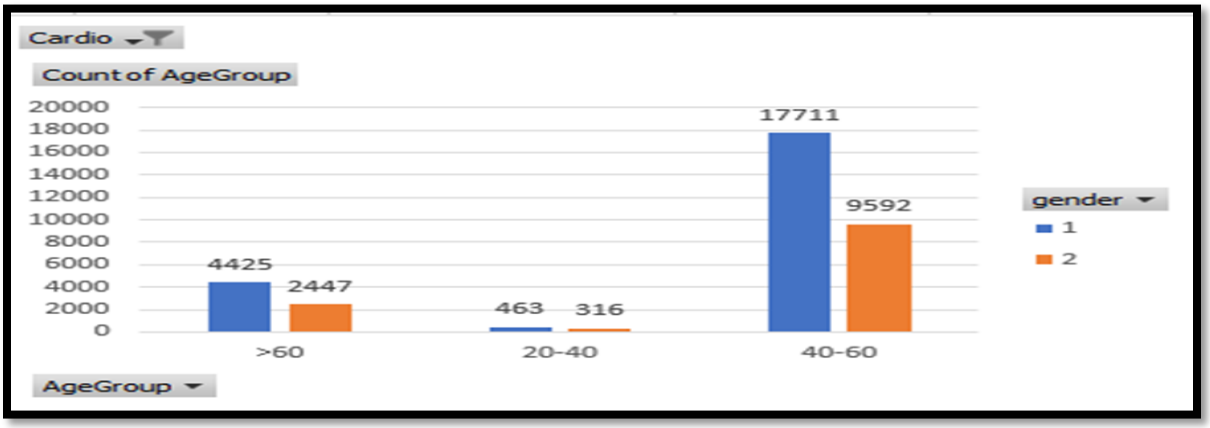

Fig. 44. Age wise distribution of males and females in the dataset

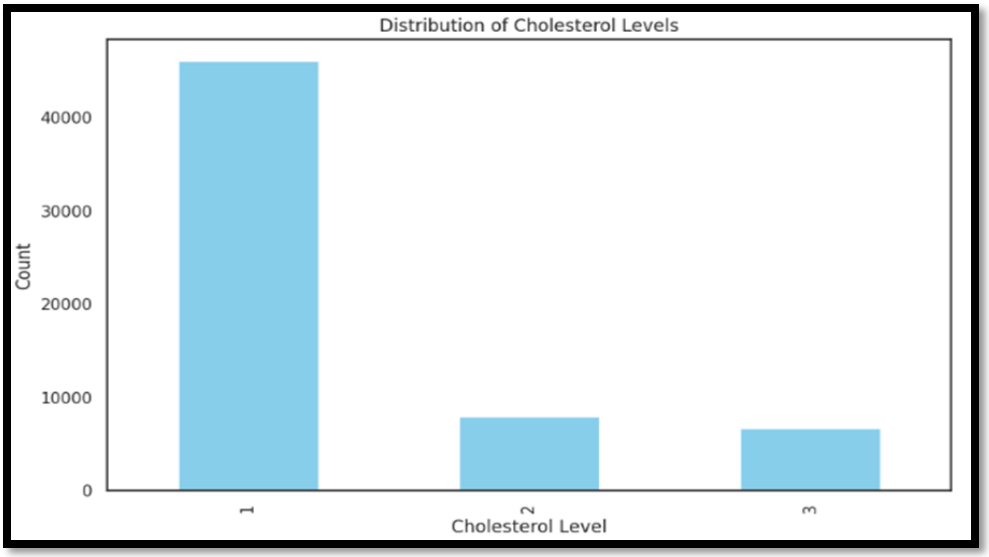

Fig. 45. bar plot showing distribution of cholesterol levels and result

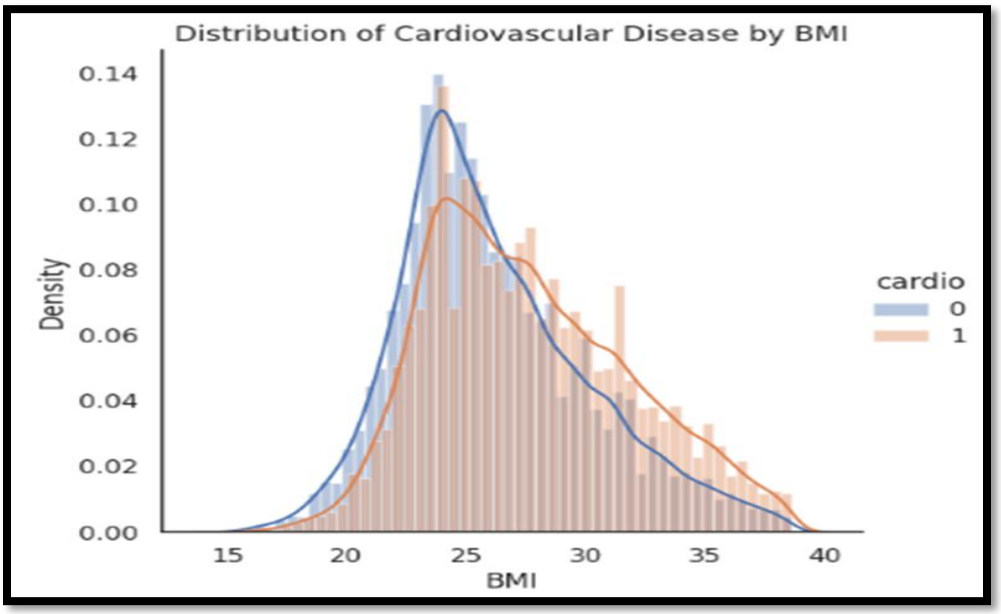

Fig. 46. Distribution of cardiovascular disease by BMI

Obese people with BMI from 30 to 40 showed greater incidence of cardiovascular disease.

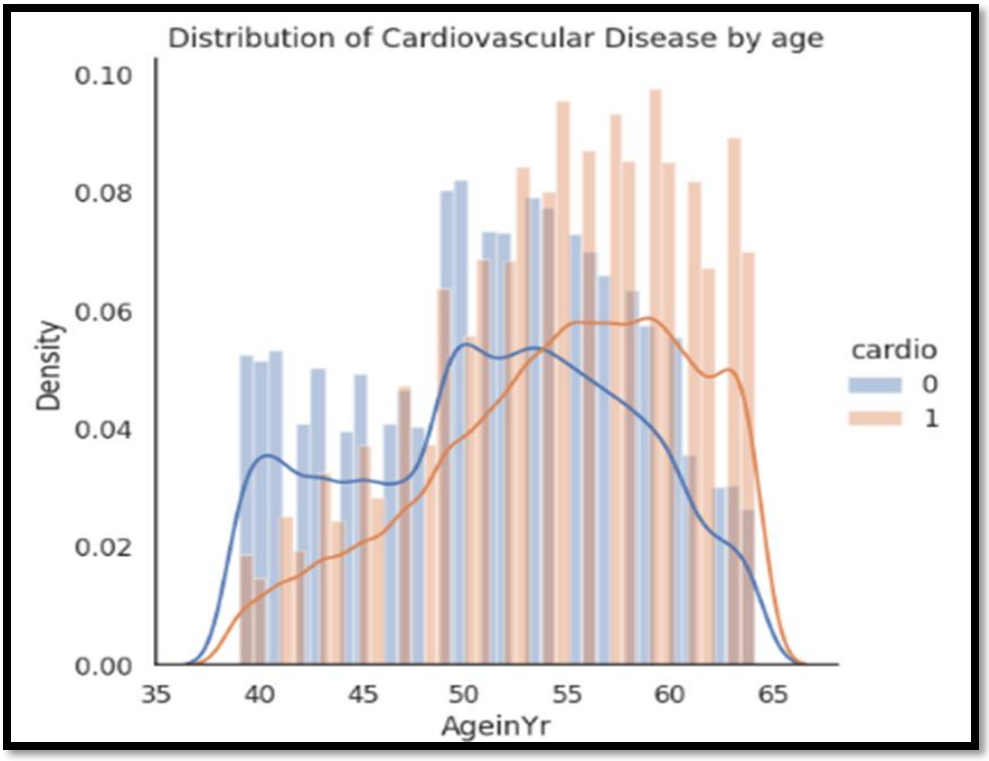

Fig. 47. Distribution of cardiovascular disease by age

The population between 50 to 60 years showed high cardiovascular disease.

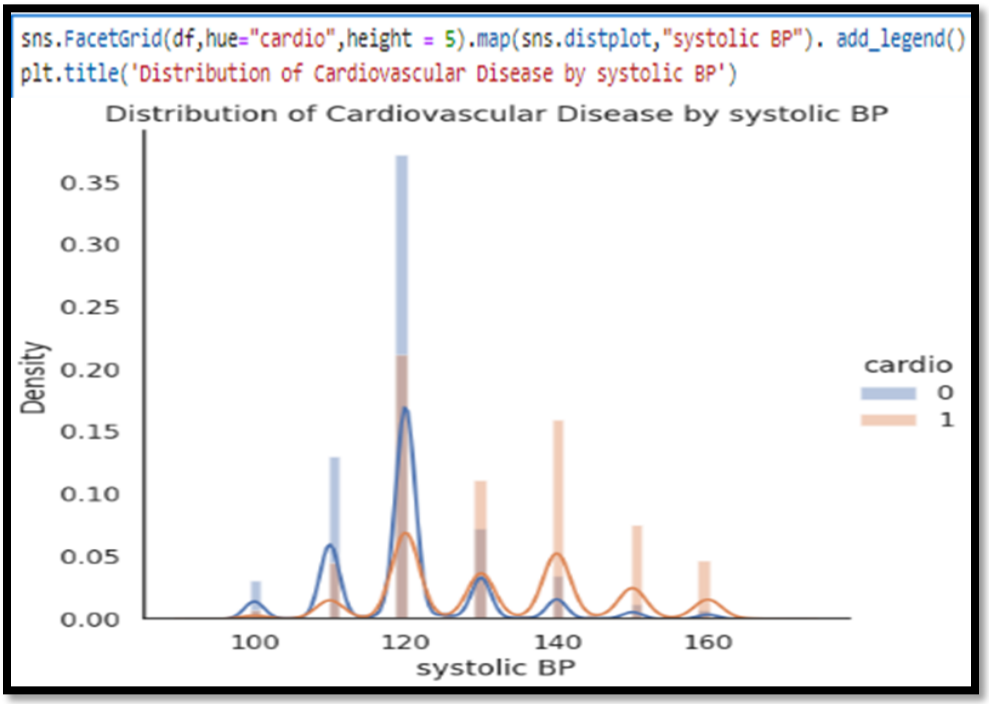

Fig. 48. Distribution of cardiovascular disease by systolic blood pressure

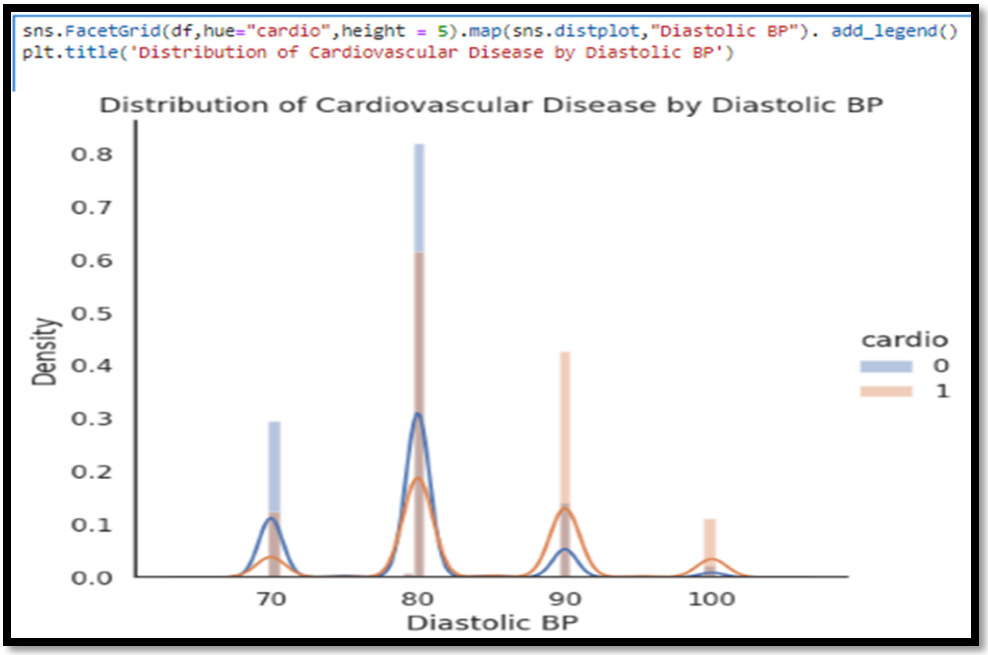

Fig. 49. Distribution of cardiovascular disease by diastolic blood pressure

The people with pre-hypertension (systolic BP 135 and above and diastolic BP 85 and above) showed higher incidence of cardiovascular diseases.

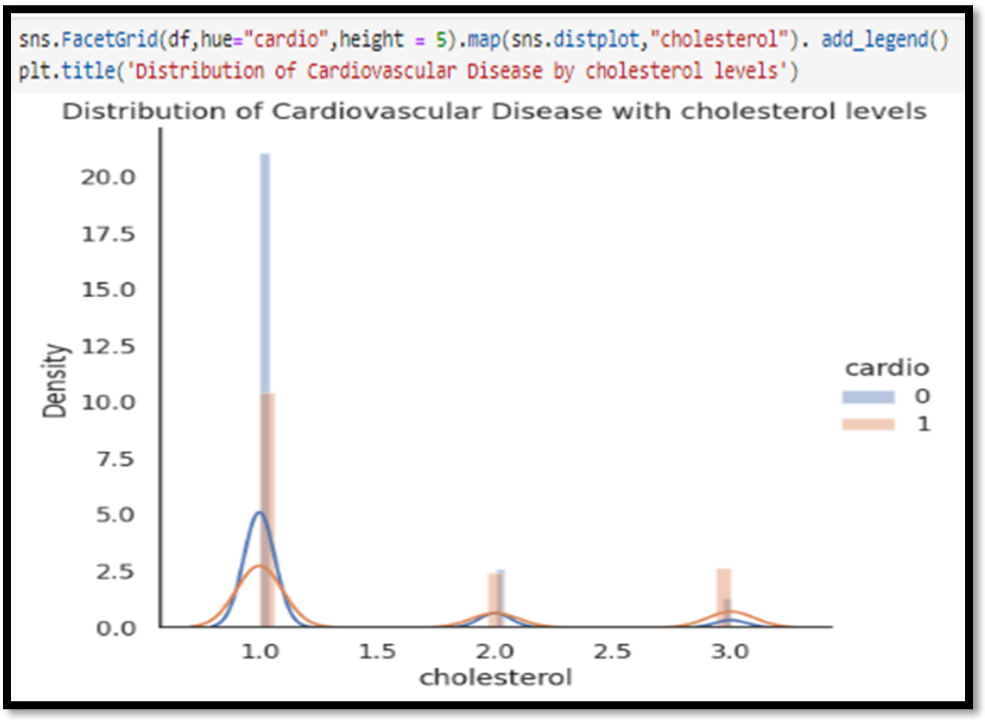

Fig. 50. Distribution of cardiovascular disease by cholesterol levels.

People with higher cholesterol levels showed high cardiovascular disease risk.

```
# Create a cross-tabulation of the "active" and "cardio" variables
active_vs_cardio = pd.crosstab(index=df["active"], columns=df["cardio"])

# Create a bar plot to compare the distribution of the "active" variable by the "cardio" variable
active_vs_cardio.plot(kind="bar", figsize=(8, 8), stacked=True)
plt.title('Distribution of Cardiovascular Disease by Physical Activity')
plt.xlabel('Physical Activity')
plt.ylabel('Count')
plt.show()
```

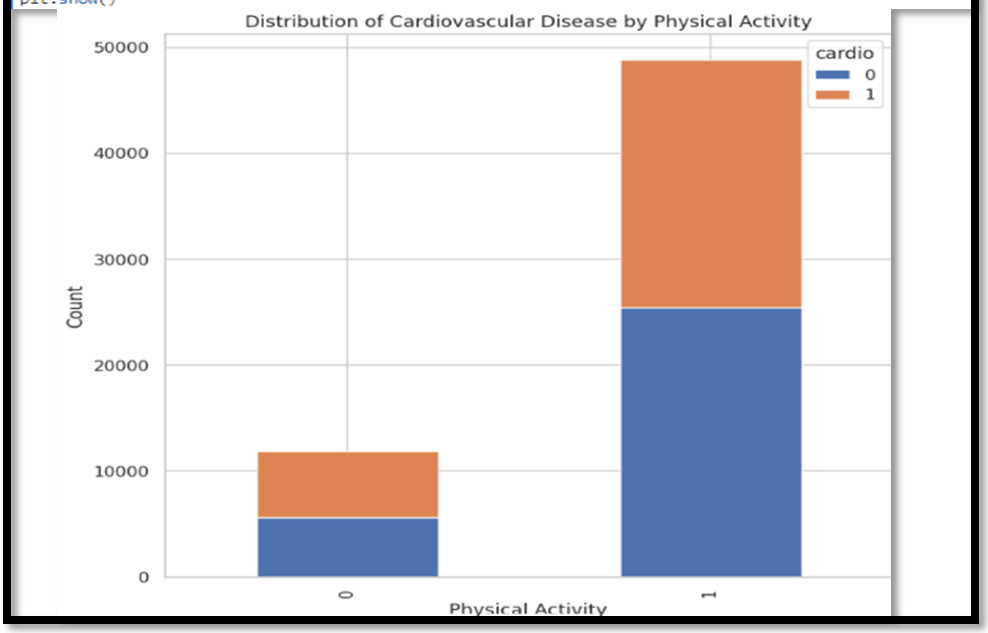

Fig. 51. Distribution of cardiovascular disease by physical activity

No significant difference observed in cardiovascular disease based on physical activity.

7. Predictive modeling techniques to evaluate the significance of the relationship between the health parameters and CVD risk

The health screening dataset contains various features such as age, gender, height, weight, blood pressure, cholesterol level, glucose level, smoking status, alcohol intake, physical activity, and the presence of cardiovascular disease. Given the nature of the dataset, several machine learning models can be used for analysis, including logistic regression, decision trees, random forests, and support vector machines (SVM).

7.1 Preprocessing the data:

Preprocess the data by handling missing values, encoding categorical variables, and scaling the features if necessary. In the below code, we first load the dataset and then preprocess it by splitting it into training and testing sets and performing feature scaling. After that, we train the logistic regression, decision tree, random forest, and support vector machine models and evaluate their performance using accuracy, precision, recall, and F1 score.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Split the data into features and target variable
X = df.drop(['id', 'cardio'], axis=1)
y = df['cardio']

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

# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Fig. 52. Code for data pre-processing

7.2 Logistic Regression:

It is widely used for binary classification tasks, such as predicting the presence of a disease based on health screening data. Logistic regression models the probability of a binary outcome based on one or more predictor variables.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# Define features and target variable
X = df[['age', 'gender', 'height', 'weight', 'systolic BP', 'Diastolic BP', 'cholesterol', 'gluc', 'smoke', 'alco', 'active']]
y = df['cardio']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize the model
model = LogisticRegression()

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

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
# Print the performance metrics
print("Logistic Regression Performance Metrics:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Logistic Regression Performance Metrics:
Accuracy: 0.7226481985324429
Precision: 0.7518407960199005
Recall: 0.6408821034775233
F1 Score: 0.6919413919413919
```

Fig. 53. Code for Logistic regression

7.3 Decision Trees:

Decision trees are used to understand the most prominent features for predicting health conditions. They are easy to interpret and can handle both numerical and categorical data.

```
#Decision Tree:
from sklearn.tree import DecisionTreeClassifier

# Initialize the model
model = DecisionTreeClassifier()

# Train the model
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the performance metrics
print("Decision Tree Performance Metrics:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Decision Tree Performance Metrics:
Accuracy: 0.624618682496496
Precision: 0.6158094580600622
Recall: 0.6052586938083121
F1 Score: 0.6104884934553855
```

Fig. 54. Code for Decision tree model

7.4 Random Forests:

Random forests are an ensemble learning method that can be used for classification tasks in health screening datasets. They are robust to overfitting and can handle large datasets with high dimensionality.

```python
#Random Forest:
from sklearn.ensemble import RandomForestClassifier

# Initialize the model
model = RandomForestClassifier()

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

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the performance metrics
print("Random Forest Performance Metrics:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Random Forest Performance Metrics:
Accuracy: 0.7016242064473576
Precision: 0.7029967891544774
Recall: 0.6685326547921968
F1 Score: 0.685331710286062
```

Fig. 54. Code for random forest model

7.5 Support Vector Machines (SVM):

SVMs are effective for binary classification tasks, including predicting the presence of a disease based on health screening features. They work well in high-dimensional spaces and are effective when the number of samples is less than the number of dimensions.

```python
#Support Vector Machine:
from sklearn.svm import SVC

# Initialize the model
model = SVC()

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

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the performance metrics
print("Support Vector Machine Performance Metrics:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Support Vector Machine Performance Metrics:
Accuracy: 0.726358314782752
Precision: 0.7629644752960392
Recall: 0.6339270568278201
F1 Score: 0.6924858704716019
```

Fig. 55. Code for support vector machine model

7.6 Evaluating the model performance:

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.722648	0.751841	0.640882	0.691941
Decision Tree	0.624124	0.615811	0.602545	0.609106
Random Forest	0.705169	0.707387	0.670908	0.688664
Support Vector Machine	0.726358	0.762964	0.633927	0.692486

Fig. 56. Evaluation of machine learning model performance based on metrics

Accuracy: It measures the proportion of correctly predicted instances out of the total instances. The Support Vector Machine model has the highest accuracy of 0.726358, followed by Logistic Regression with an accuracy of 0.722648.

Precision: It measures the proportion of true positive predictions out of the total positive predictions. The Support Vector Machine model also has the highest precision of 0.762964, indicating its ability to avoid false positives.

Recall: It measures the proportion of true positive predictions out of the actual positives. The Random Forest model has the highest recall of 0.670908, indicating its ability to identify all actual positive instances.

F1 Score: The Support Vector Machine model has the highest F1 score of 0.692486, which is a balance between precision and recall.

Based on these metrics, the Support Vector Machine model appears to perform the best across multiple evaluation metrics. However, the choice of the best model ultimately depends on the specific requirements and trade-offs of the problem at hand. It's important to consider the specific goals and constraints of the application when selecting the best model.

8. Discussion:

Our analysis aimed to identify key health indicators associated with cardiovascular disease risk and develop predictive models using machine learning techniques. The results reveal several significant findings that have implications for cardiovascular disease prevention and management.

We have cleaned our dataset and checked for any null values in the dataset. As part of our initial data exploration, box plots and scatter plots were utilized to visually identify outliers in attributes like systolic blood pressure, weight, cholesterol levels and BMI. We employed standard approaches including z-scores (values exceeding +/- 3 standard deviations from mean) and the interquartile range rule (outside 1.5 times the IQR) to detect outliers. To manage outlier effects, identified extreme values were replaced with the median values.

Then, we performed descriptive statistics. The descriptive statistical analysis of the health screening dataset provides valuable insights into the cardiovascular risk profile of the study population. Firstly, the sample constitutes predominantly middle-aged individuals with a mean age of 52.87 years. There is male preponderance, indicative of the higher incidence of cardiovascular disease among men in this age group. The parameters also reveal a cohort at elevated CVD risk based on attributes like higher mean BMI of 26.86 kg/m2 falling into the overweight category, widespread hypertension evidenced through mean systolic and diastolic blood pressures of 126.05 mmHg and 81.59 mmHg respectively, and borderline high mean total cholesterol levels of 1.35 mmol/L.

In contrast, the dataset showed relatively low smoking and alcohol consumption patterns, with only 8.8% identified as smokers and 5.2% as alcohol users. This points out that other risk factors play a more central role in driving cardiovascular risk in the sample. The summations of height, weight and blood glucose metrics offer added descriptive depth into population characteristics.

In total, the descriptive statistics portray a sample with escalated risk factor burdens indicative of cardiovascular vulnerability. The prevalence of concerning attributes like above-normal BMI and cholesterol, hypertension and prediabetes forewarns of the possibility of adverse cardiovascular outcomes in the group unless risk mitigation interventions are implemented.

The normality test included the Shapiro-Wilk test. By performing the normality test we inferred that our data was not normally distributed. The correlation matrix provided critical insights into the interrelationships between key health indicators and cardiovascular disease risk and presence within the dataset. Several notable associations stand out.

Firstly, strong positive correlation is observed between age and attributes like systolic and diastolic blood pressure, cholesterol levels, and BMI. This aligns with typical aging trajectories where these cardiovascular risk factors deteriorate with advancing age. Age also shows modest positive correlation to cardiovascular disease presence, indicative of age-related accumulation of vascular damage.

Blood pressure parameters, especially systolic BP, also correlate strongly with body weight and BMI, along expected lines where excess weight escalates hypertension risk. Hypertension marked through high systolic and diastolic pressures is associated with further elevated cardiovascular risk.

While smoking and alcohol consumption correlate weakly with gender, indicating lifestyle disease exposure among males, they show less direct association to cardiovascular outcome. This implies other non-behavioral factors like cholesterol, BMI and age may be more predictive in our dataset.

Complementing the Pearson correlation analysis, we also employed non-parametric Kendall's tau and Spearman's rank correlation approaches to evaluate monotonic relationships between key variable pairs in our dataset.

The outputs reinforce our earlier observations. Kendall's tau correlation coefficient of 0.47 and Spearman's rho value of 0.58 indicate a moderately positive association between age and systolic blood pressure ranks. This aligns with typical worsening of hypertension with progressing age. The very low p-values suggest these correlations are highly statistically significant, allowing us to reject the null hypothesis of no correlation between the attribute ranks.

Similarly, we notice a weak but significant negative correlation between smoking status and cholesterol levels per the rank-based analysis. This requires further investigation to interpret but may reflect dietary and lifestyle variations among smokers influencing observed cholesterol levels.

The concordance between the Pearson and non-parametric correlation techniques boosts overall confidence in the relationships detected between variables in the dataset. By accounting for potential non-normal distributions through rank transformation, Kendall and Spearman analysis provides additional confirmation regarding important risk factor associations and their implications for cardiovascular disease in the cohort.

We also performed a chi-square test. From the results, we inferred that there is a significant relationship between physical activity, smoking, cholesterol, glucose, alcoholism and cardiovascular disease. Based on the nature of the dependent variable (categorical, dichotomous) in the dataset, we could not perform the Mann Whitney U test, Kruskal Wallis Test, Wilcoxon Signed Rank test on our dataset.

The binary logistic regression model provides several meaningful insights into the predictive relationships between key health indicators and cardiovascular disease likelihood in our cohort.

The regression analysis confirms the significant association of variables like age, blood pressure, cholesterol, blood glucose, smoking, alcohol use and physical activity to cardiovascular disease, even after adjusting for other covariates. The directionality of coefficients aligns with clinical expectations - for example systolic BP, cholesterol and advancing age seem associated with higher CVD probability while physical activity depicts a protective effect.

The odds ratios quantify the magnitude of risk factor effects. Variables showing strongest influence are systolic BP (OR 1.06), age (OR 1.08) and cholesterol (OR 1.41). This knowledge helps inform priorities for preventive intervention, where managing raised blood pressure, dyslipidemia and encouraging active lifestyles for aging populations would have maximum impact on lowering cardiovascular risk at the cohort and population levels.

However, the logistic model has limitations in predictive performance as shown through the modest Pseudo R2 value of 0.186. This implies meaningful variability in cardiovascular outcomes remains unexplained and warrants exploration into additional risk factors.

For training machine learning models to our dataset, we have first pre-processed the data. We split the dataset into 80% training set and 20% testing set. We trained our data for 4 machine learning models, they are Logistics regression, Decision Tree, Random forests, and Support Vector Machine models. We used accuracy, precision, recall and F1 score as metrics to find the best fit model for our dataset. The Support Vector Machine (SVM) model performed best with accuracy of 73%. The models need optimization through techniques like managing class imbalance. More refined features and a larger dataset can also improve model robustness.

An important discussion point is the non-normal distribution of certain variables like cholesterol levels and blood glucose values in our dataset. This skewness may have introduced bias during analysis. Applying normalization techniques can mitigate such effects. Though commonly used, our reliance on accuracy as the evaluation metric also has drawbacks. Including metrics like F1-scores and AUC-ROC plots can enhance model comparison.

In summary, our analysis provides valuable insights on the feature importance of cardiovascular disease while demonstrating the promise of machine learning for predictive analytics in healthcare. But it also highlights opportunities to refine data collection, feature engineering, model tuning and evaluation metrics. Pursuing these can

aid the development of robust screening and prognostic tools for dealing with cardiovascular disease burden. Our study provides the foundation to progress in that direction through more rigorous data science approaches.

9. Summary:

Based on the results of our correlation and hypothesis testing analysis, we concluded that age, gender, blood pressure, cholesterol level, and smoking status are the most significant risk factors for cardiovascular disease in our study population. We rejected the null hypothesis through evidence of strong correlations between cardiovascular disease and attributes like systolic blood pressure, age, cholesterol levels, weight, and BMI. In contrast, variables like blood glucose, smoking status, alcohol intake and physical activity exhibited weaker correlations.

These findings validate established research on the major risk factors and predictors for cardiovascular disease onset and progression. We can confirm that addressing modifiable risk factors including obesity, hypertension, hyperlipidemia and smoking habits at the patient and population level will have a high impact on decreasing overall cardiovascular disease burden.

The machine learning models we developed, including logistic regression, random forests and support vector machines, utilize these key health indicators to predict the presence of cardiovascular disease with decent accuracy. There is certainly scoped to improve model robustness through larger and better-quality datasets, tuning model hyperparameters and managing class imbalance. But overall, our exploratory study provides valuable insights and direction to apply predictive analytics in screening, diagnosing and prognosticating cardiovascular disease in clinical practice.

Our analysis established evidence for age, gender, blood pressure, cholesterol and smoking status as the strongest risk factors for cardiovascular disease. It provides real-world data to motivate policies and interventions targeted at managing modifiable risk factors like hypertension, obesity and high cholesterol to mitigate surging cardiovascular disease prevalence. It also demonstrated the viability of predictive modeling to enable personalized cardiovascular disease risk estimation and stratified management at scale. Further research should explore the causal mechanisms behind these observations and develop optimal risk prediction tools for routine clinical use.

10. Limitations:

This study has certain limitations that impact the interpretation and generalizability of the findings. The cross-sectional design provides insights on predictor-outcome associations at a single time point but cannot ascertain temporality or causal relationships. We cannot definitively state whether variables like hypertension, obesity or dyslipidemia precede cardiovascular disease onset or result from its progression.

Additionally, the quality and accuracy of health screening data remains a constraint, despite our best efforts at data cleaning and preprocessing. Issues like measurement errors, missing values and reporting subjectivity can distort observations. The sampling methodology may also introduce selection bias, restricting the wider applicability of findings. We cannot fully ensure the sample represents the population of interest. Other limitations include the modest sample size, which impacts the statistical power to detect weaker associations in the dataset.

Strengthening the limitations of this exploratory study would require mitigating these study design, data and sampling constraints through more robust techniques. Adopting a longitudinal cohort design tracking subjects over time can establish temporality between predictors and the cardiovascular disease outcome. Improving health screening data rigor through steps like cross-verification of machine readings, multiple imputation for missing variables and outlier management would enhance validity. Employing probability sampling methodology would also make the sample more representative of the target population.

While keeping these limitations in perspective, our study nonetheless provides actionable and directional insights on major risk factors for rising cardiovascular disease burden. The associations found between attributes like blood pressure, cholesterol and cardiovascular disease risk align with existing clinical evidence. Our analysis sets up

further investigations into these relationships through stronger research approaches. Managing limitations through steps outlined above can help increase the utility and contribution of similar health screening data-based explorations to inform health policy and planning.

References

Acevedo, M., Varleta, P., Casas-Cordero, C., Berríos, A., Navarrete, C., & López, R. (2021). Prevalence and determinants of ideal cardiovascular health in a latin women cohort: A cross-sectional study. The Lancet Regional Health - Americas, 4, 100071. https://doi.org/10.1016/j.lana.2021.100071

Akinosun, A. S., Kamya, S., Watt, J., Johnston, W., Leslie, S. J., & Grindle, M. (2023). Cardiovascular disease behavioural risk factors in rural interventions: Cross-sectional study. Scientific Reports, 13(1), 13376. https://doi.org/10.1038/s41598-023-39451-5

Bischops, A.C., De Neve, JW., Awasthi, A. et al. A cross-sectional study of cardiovascular disease risk clustering at different socio-geographic levels in India. Nat Commun 11, 5891 (2020). https://doi.org/10.1038/s41467-020-19647-3

Islami, F., Mańczuk, M., Vedanthan, R., Vatten, L., Polewczyk, A., Fuster, V., Boffetta, P., & Zatoński, W. A. (2011). A cross-sectional study of cardiovascular disease and associated factors. Annals of agricultural and environmental medicine : AAEM, 18(2), 255–259.

Shrivastava, S. R., Ghorpade, A. G., & Shrivastava, P. S. (2015). A Community-based Cross-sectional Study of Cardiovascular Risk in a Rural Community of Puducherry. Heart views : the official journal of the Gulf Heart Association, 16(4), 131–136. https://doi.org/10.4103/1995-705X.172195

Appendix:

Data Importing:

1. IMPORTING AND EXPLORING DATASET

```
'''we have imported the dataset into pandas datafrane and assigned it to variable df. We used shape function to read the number of columns an
info function to read the data types of the values in all the columns. All per the below result, BMICat has non-numerical and AgeGroup variab
There are no missing values in the columns'''

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#Load data
file_path = 'Health Screening Data.csv'
df = pd.read_csv(file_path)
#explore data
print ('Data shape:',df.shape)
print (df.info())
```

1.
```
df = df.rename(columns={'ap_hi': 'systolic BP', 'ap_lo': 'Diastolic BP'})
df.columns
```

2.
```
'''head function is used to get the values in the first five rows of columns'''
df.head()
```

3.
```
'''dtypes function used to read the data types of the values in the columns'''
df.dtypes
```

4.
```
'''describe function is used to review the summary statistics of the dataset'''
df.describe().T
```

5.

Data Cleaning:

2. CLEANING THE DATA

6.
```
df.isnull().sum()
```

7.
```
df.columns
```

Outliers identification and visualization:

8.
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Remove non-numerical and categorical variables
non_numerical_columns = ['Unnamed','id','gender', 'smoke','alco','active','cholesterol','gluc','cardio','BMICat', 'AgeGroup']
numerical_columns = [col for col in df if col not in non_numerical_columns]

num_plots = len(numerical_columns)
num_cols = 2  # Number of columns for subplots
num_rows = (num_plots + num_cols - 1) // num_cols  # Calculate the number of rows needed

plt.figure(figsize=(15, 10))  # Set the overall figure size
colors = ['skyblue', 'salmon', 'lightgreen', 'lightcoral']  # Define colors for the boxplots
for i, column in enumerate(numerical_columns):
    plt.subplot(num_rows, num_cols, i+1)  # Create subplots
    sns.boxplot(x=df[column], color=colors[i % len(colors)])  # Set color for each boxplot
    plt.title(f'Boxplot for {column}')
plt.tight_layout()  # Adjust the layout
plt.show()
```

visualization of outliers using scattered plot method: To identify outliers using a scatter plot, you can visually inspect the data for any points that deviate significantly from the overall pattern of the plot. In the health screening dataset, you can create a scatter plot to visualize the relationship between two variables and identify potential outliers.

9.
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Set the aesthetic style of the plots
sns.set(style="white")

# Create scattered plots for numerical variables in subplots
# Remove non-numerical and categorical variables
non_numerical_columns = ['id','gender', 'smoke','alco','active','cholesterol','gluc','cardio','BMICat', 'AgeGroup']  # Replace with actual nc
numerical_columns = [col for col in numerical_columns if col not in non_numerical_columns]
num_plots = len(numerical_columns)
num_cols = 3  # Number of columns for subplots
num_rows = (num_plots + num_cols - 1) // num_cols  # Calculate the number of rows needed

plt.figure(figsize=(18, 12))  # Set the overall figure size
for i, column in enumerate(numerical_columns):
    plt.subplot(num_rows, num_cols, i+1)  # Create subplots
    sns.scatterplot(x=df.index, y=df[column], alpha=0.7, s=100, color=colors[i % len(colors)])  # Set color for each scatter plot and adjust
    #sns.scatterplot(x=data[column], color=colors[i % len(colors)])  # Set color for each boxplot
    plt.title(f'Scatterplot for {column}')
    plt.grid(False)  # Remove grid lines
plt.tight_layout()  # Adjust the layout
plt.show()
```

10.
```
# Import necessary libraries
import pandas as pd

# Select numerical columns
numerical_columns = df.select_dtypes(include=['number'])

# Calculate the IQR for numerical columns
Q1 = numerical_columns.quantile(0.25)
Q3 = numerical_columns.quantile(0.75)
IQR = Q3 - Q1

# Identify outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = (numerical_columns< lower_bound) | (numerical_columns> upper_bound)

# Print the outliers for each variable
print(outliers.sum())
```

Handling outliers:

4. HANDLING OUTLIERS: To handle outliers in the provided health screening dataset, you can use various techniques such as removing outliers, transforming the data, imputing with central tendencies, binning, and using robust statistical methods. Here's how you can perform these tasks using Python:

```
'''5. Using Robust Statistical Methods'''
#Using median and percentile-based statistics
median = df['AgeinYr'].median()
upper_quartile = df['AgeinYr'].quantile(0.75)
lower_quartile = df['AgeinYr'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['AgeinYr'] > lower_bound) & (df['AgeinYr'] < upper_bound)]

median = df['height'].median()
upper_quartile = df['height'].quantile(0.75)
lower_quartile = df['height'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['height'] > lower_bound) & (df['height'] < upper_bound)]

median = df['weight'].median()
upper_quartile = df['weight'].quantile(0.75)
lower_quartile = df['weight'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['weight'] > lower_bound) & (df['weight'] < upper_bound)]

median = df['systolic BP'].median()
upper_quartile = df['systolic BP'].quantile(0.75)
lower_quartile = df['systolic BP'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['systolic BP'] > lower_bound) & (df['systolic BP'] < upper_bound)]
```

11.
```
median = df['Diastolic BP'].median()
upper_quartile = df['Diastolic BP'].quantile(0.75)
lower_quartile = df['Diastolic BP'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['Diastolic BP'] > lower_bound) & (df['Diastolic BP'] < upper_bound)]

median = df['BMI'].median()
upper_quartile = df['BMI'].quantile(0.75)
lower_quartile = df['BMI'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['BMI'] > lower_bound) & (df['BMI'] < upper_bound)]
```

12.

Visualization of outliers after handling:

visualization of outliers using scattered plot method: To identify outliers using a scatter plot, you can visually inspect the data for any points that deviate significantly from the overall pattern of the plot. In the health screening dataset, you can create a scatter plot to visualize the relationship between two variables and identify potential outliers.

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

# Set the aesthetic style of the plots
sns.set(style="white")

# Create scattered plots for numerical variables in subplots
# Remove non-numerical and categorical variables
non_numerical_columns = ['id','gender', 'smoke','alco','active','cholesterol','gluc','cardio','BMICat', 'AgeGroup']  # Replace with actual nc
numerical_columns = [col for col in numerical_columns if col not in non_numerical_columns]
num_plots = len(numerical_columns)
num_cols = 3  # Number of columns for subplots
num_rows = (num_plots + num_cols - 1) // num_cols  # Calculate the number of rows needed

plt.figure(figsize=(18, 12))  # Set the overall figure size
for i, column in enumerate(numerical_columns):
    plt.subplot(num_rows, num_cols, i+1)  # Create subplots
    sns.scatterplot(x=df.index, y=df[column], alpha=0.7, s=100, color=colors[i % len(colors)])  # Set color for each scatter plot and adjust
    #sns.scatterplot(x=data[column], color=colors[i % len(colors)])  # Set color for each boxplot
    plt.title(f'Scatterplot for {column}')
    plt.grid(False)  # Remove grid lines
plt.tight_layout()  # Adjust the layout
plt.show()
```

13.

14.
```
# Import necessary libraries
import pandas as pd

# Select numerical columns
numerical_columns = df.select_dtypes(include=['number'])

# Calculate the IQR for numerical columns
Q1 = numerical_columns.quantile(0.25)
Q3 = numerical_columns.quantile(0.75)
IQR = Q3 - Q1

# Identify outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = (numerical_columns< lower_bound) | (numerical_columns> upper_bound)

# Print the outliers for each variable
print(outliers.sum())
```

4. HANDLING OUTLIERS: To handle outliers in the provided health screening dataset, you can use various techniques such as removing outliers, transforming the data, imputing with central tendencies, binning, and using robust statistical methods. Here's how you can perform these tasks using Python:

15.
```
'''5. Using Robust Statistical Methods'''
#Using median and percentile-based statistics
median = df['AgeinYr'].median()
upper_quartile = df['AgeinYr'].quantile(0.75)
lower_quartile = df['AgeinYr'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['AgeinYr'] > lower_bound) & (df['AgeinYr'] < upper_bound)]

median = df['height'].median()
upper_quartile = df['height'].quantile(0.75)
lower_quartile = df['height'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['height'] > lower_bound) & (df['height'] < upper_bound)]

median = df['weight'].median()
upper_quartile = df['weight'].quantile(0.75)
lower_quartile = df['weight'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['weight'] > lower_bound) & (df['weight'] < upper_bound)]

median = df['systolic BP'].median()
upper_quartile = df['systolic BP'].quantile(0.75)
lower_quartile = df['systolic BP'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['systolic BP'] > lower_bound) & (df['systolic BP'] < upper_bound)]
```

16.
```
median = df['Diastolic BP'].median()
upper_quartile = df['Diastolic BP'].quantile(0.75)
lower_quartile = df['Diastolic BP'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['Diastolic BP'] > lower_bound) & (df['Diastolic BP'] < upper_bound)]

median = df['BMI'].median()
upper_quartile = df['BMI'].quantile(0.75)
lower_quartile = df['BMI'].quantile(0.25)
iqr = upper_quartile - lower_quartile
upper_bound = upper_quartile + 1.5 * iqr
lower_bound = lower_quartile - 1.5 * iqr
df = df[(df['BMI'] > lower_bound) & (df['BMI'] < upper_bound)]
```

5. Outliers visualization after handling outliers

17.
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Remove non-numerical and categorical variables
non_numerical_columns = ['id','gender', 'smoke','alco','active','cholesterol','gluc','cardio','BMICat', 'AgeGroup']
numerical_columns = [col for col in numerical_columns if col not in non_numerical_columns]

num_plots = len(numerical_columns)
num_cols = 2  # Number of columns for subplots
num_rows = (num_plots + num_cols - 1) // num_cols  # Calculate the number of rows needed

plt.figure(figsize=(15, 10))  # Set the overall figure size
colors = ['skyblue', 'salmon', 'lightgreen', 'lightcoral']  # Define colors for the boxplots
for i, column in enumerate(numerical_columns):
    plt.subplot(num_rows, num_cols, i+1)  # Create subplots
    sns.boxplot(x=df[column], color=colors[i % len(colors)])  # Set color for each boxplot
    plt.title(f'Boxplot for {column}')
plt.tight_layout()  # Adjust the layout
plt.show()
```

18.
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Set the aesthetic style of the plots
sns.set(style="white")

# Create scattered plots for numerical variables in subplots
# Remove non-numerical and categorical variables
non_numerical_columns = ['id','gender', 'smoke','alco','active','cholesterol','gluc','cardio','BMICat', 'AgeGroup']
numerical_columns = [col for col in numerical_columns if col not in non_numerical_columns]
num_plots = len(numerical_columns)
num_cols = 3  # Number of columns for subplots
num_rows = (num_plots + num_cols - 1) // num_cols  # Calculate the number of rows needed

plt.figure(figsize=(18, 12))  # Set the overall figure size
for i, column in enumerate(numerical_columns):
    plt.subplot(num_rows, num_cols, i+1)  # Create subplots
    sns.scatterplot(x=df.index, y=df[column], alpha=0.7, s=100, color=colors[i % len(colors)])  # Set color for each
    #sns.scatterplot(x=data[column], color=colors[i % len(colors)])  # Set color for each boxplot
    plt.title(f'Scatterplot for {column}')
    plt.grid(False)  # Remove grid lines
plt.tight_layout()  # Adjust the layout
plt.show()
```

19.
```
'''describe.T will transpose the describe function result'''
df.describe().T
```

Relationship analysis between variables – correlation matrix, Kendall's correlation and Spearman's rank correlation:

5. RELATIONSHIP ANALYSIS

20.
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
numerical_data = df.select_dtypes(include=['int64', 'float64'])
correlation_matrix = numerical_data.corr()
plt.figure(figsize=(10, 8))
heatmap = sns.heatmap(correlation_matrix, annot=True, square = True, fmt = '.2f', annot_kws={'size':10},cmap="coolwarm")
plt.title('Correlation between risk factors and cardiovascular disease')
plt.show()
print (correlation_matrix)
```

21.
```
import pandas as pd
from scipy.stats import kendalltau
import matplotlib.pyplot as plt
import seaborn as sns

# Select the independent variables and the dependent variable
independent_vars = ['height', 'weight', 'cholesterol','systolic BP', 'Diastolic BP', 'AgeinYr', 'BMI','smoke', 'alco','active']
dependent_var = 'cardio'  # Assuming 'cardio' is the categorical dependent variable

# Calculate Kendall's tau coefficient for each independent variable
correlations = {}
for var in independent_vars:
    tau, p_value = kendalltau(df[var], df[dependent_var])
    correlations[var] = tau

# Visualize the result
plt.figure()
plt.bar(correlations.keys(), correlations.values())
plt.xlabel('Independent Variable')
plt.ylabel("Kendall's Tau Coefficient")
plt.title("Kendall's Tau Coefficient for Each Independent Variable")
plt.xticks(rotation=45)  # Rotate the x-axis labels for clarity
plt.show()

# Print the result
print("Kendall's rank correlation coefficient:", tau)
print("P-value:", p_value)
```

22.
```
import pandas as pd
from scipy.stats import spearmanr
import seaborn as sns
import matplotlib.pyplot as plt


# Select the independent continuous variables and the dichotomous dependent variable
independent_vars = ['height', 'weight', 'cholesterol','systolic BP', 'Diastolic BP', 'AgeinYr', 'BMI','smoke', 'alco','active']
dependent_var = 'cardio'  # Assuming 'cardio_disease' is the dichotomous dependent variable

# Calculate Spearman's rank correlation for each independent variable
correlations = {}
for var in independent_vars:
    rho, p_value = spearmanr(df[var], df[dependent_var])
    correlations[var] = rho

# Visualize the correlations
plt.figure()
plt.bar(correlations.keys(), correlations.values())
plt.xlabel('Independent Variable')
plt.ylabel("Spearman's Rank Correlation Coefficient")
plt.title("Spearman's Rank Correlation for Each Independent Variable")
plt.xticks(rotation=45)  # Rotate the x-axis labels for clarity
plt.show()

# Print the result
print("Spearman's Rank Correlation Coefficient:", rho)
print("P-value:", p_value)
```

Normality testing of the data:

23.

```python
#Shapiro-Wilk test:
from scipy import stats
import numpy as np
from scipy.stats import shapiro
#Convert all columns to numeric (ignoring non-numeric values)
df = df.apply(pd.to_numeric, errors='coerce')

# Drop columns with less than 3 non-NaN values
df = df.dropna(thresh=3, axis=1)

# Check if there are enough data points
if len(df) < 3:
    print("Insufficient data points for normality test.")
else:
    # Flatten the DataFrame into a one-dimensional array
    data = df.values.flatten()

    # Perform Shapiro-Wilk test
    statistic, p_value = shapiro(data)

    # Print the results
    print(f'Statistic: {statistic}, p-value: {p_value}')

    # Check the p-value against a significance level (e.g., 0.05)
    if p_value > 0.05:
        print("The dataset appears to be normally distributed.")
    else:
        print("The dataset does not appear to be normally distributed.")
```

24.

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm


# Select numerical columns excluding specific categorical variables
numerical_data = df.select_dtypes(include=['number']).drop(columns=['id', 'gender', 'smoke', 'alco', 'active', 'cholesterol', 'gluc', 'cardio

# Plot QQ-Plots for selected numerical variables
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 15))
for i, column in enumerate(numerical_data.columns):
    ax = axes[i // 3, i % 3]
    sm.qqplot(numerical_data[column], line ='45', ax=ax)
    ax.set_title(column)
plt.tight_layout()
plt.show()
```

testing Kurtosis and Skewness of the dataset:

Skewness Skewness measures the asymmetry of the distribution. A skewness value of 0 indicates a symmetrical distribution. A negative skewness indicates that the left tail of the distribution is longer or fatter than the right tail, and a positive skewness indicates the opposite . Kurtosis Kurtosis measures the tailedness of the distribution. A kurtosis value of 3 indicates a normal distribution. Excess kurtosis (kurtosis min3) greater than 0 indicates heavier tails than a normal distribution, and less than 0 indicates lighter ai lies.

25.
```
import pandas as pd
from scipy.stats import kurtosis, skew

# Calculate the kurtosis of the dataset
data_kurtosis = df.kurtosis()
print('kurtosis of the data:')
print(data_kurtosis)

# Calculate the skewness of the dataset
data_skewness = df.skew()
print('skewness of the data:')
print(data_skewness)
```

Descriptive statistics:

Descriptive Statistics

26.
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, roc_auc_score

# Select continuous numerical variables
numerical_vars = ['age', 'height', 'weight', 'systolic BP', 'Diastolic BP', 'AgeinYr', 'BMI']

# Create subplots for each variable
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 10))

# Plot histograms for each variable
for i, var in enumerate(numerical_vars):
    row = i // 3
    col = i % 3
    axes[row, col].hist(df[var], bins=10, alpha=0.7, density=True, color='b', edgecolor='black', label='Actual Distribution')
    axes[row, col].set_title(var)

    # Add normal distribution line
    mu, sigma = df[var].mean(), df[var].std()
    xmin, xmax = df[var].min(), df[var].max()
    x = np.linspace(xmin, xmax, 100)
    p = stats.norm.pdf(x, mu, sigma)
    axes[row, col].plot(x, p, 'k', linewidth=2, label='Normal Distribution')
    axes[row, col].legend()

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

Hypothesis testing – Chi-Square test:

27.
```
import pandas as pd
from scipy.stats import chi2_contingency


# chi-square test on the 'smoke' and 'cardio' variables
contingency_table = pd.crosstab(df['smoke'], df['cardio'])

# Perform the chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Statistic:", chi2)
print("P-Value:", p)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
if p > 0.05:
    print('there is no significant relationship between smoking and cardiovascular diseasee')
else:
    print('there is Significant relationship between smoking and cardiovascular disease')
```

28.
```
import pandas as pd
from scipy.stats import chi2_contingency


# chi-square test on the 'smoke' and 'cardio' variables
contingency_table = pd.crosstab(df['gluc'], df['cardio'])

# Perform the chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Statistic:", chi2)
print("P-Value:", p)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
if p > 0.05:
    print('there is no significant relationship between glucose levels and cardiovascular disease')
else:
    print('there is a significant relationship between glucose levels and cardiovascular disease')
```

29.
```
import pandas as pd
from scipy.stats import chi2_contingency


# chi-square test on the 'smoke' and 'cardio' variables
contingency_table = pd.crosstab(df['cholesterol'], df['cardio'])

# Perform the chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Statistic:", chi2)
print("P-Value:", p)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
if p > 0.05:
    print('there is no significant relationship between cholesterol levels and cardiovascular disease')
else:
    print('there is a significant relationship between cholesterol levels and cardiovascular disease')
```

30.
```
import pandas as pd
from scipy.stats import chi2_contingency


# chi-square test on the 'smoke' and 'cardio' variables
contingency_table = pd.crosstab(df['alco'], df['cardio'])

# Perform the chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Statistic:", chi2)
print("P-Value:", p)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
if p > 0.05:
    print('there is no significant relationship between alcoholism and cardiovascular disease')
else:
    print('there is a significant relationship between alcoholism and cardiovascular disease')
```

31.
```
import pandas as pd
from scipy.stats import chi2_contingency


# chi-square test on the 'smoke' and 'cardio' variables
contingency_table = pd.crosstab(df['active'], df['cardio'])

# Perform the chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Statistic:", chi2)
print("P-Value:", p)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:", expected)
if p > 0.05:
    print('there is no significant relationship between physical activity and cardiovascular disease')
else:
    print('there is a significant relationship between physical activity and cardiovascular disease')
```

Logistic regression testing:

Kruskal-Wallis H test: (sometimes also called the "one-way ANOVA on ranks") is a rank-based nonparametric test that can be used to determine if there are statistically significant differences between two or more groups of an independent variable on a continuous or ordinal dependent variable. As our dependant variable as categorical values, this test was not performed on our dataset.

Wilcoxon Signed-Rank test This non-parametric test is used to compare the distributions of two related groups when the dependent variable is continuous and the independent variable is dichotomous. Therefore, it was not performed for our dataset.

Logistic regression model: Logistic regression is used to predict the categorical dependent variable. It's used when the prediction is categorical, for example, yes or no, true or false, 0 or 1. logistic regression is used for categorical outcome variables, such as death. Independent variables can be continuous, categorical, or a mix of both. Logistic regression model is used for our dataset as nature of the dependant variable (cardio) is categorical and binary.

32.
```
import pandas as pd
import statsmodels.api as sm


# Select the independent variables (health parameters) and the dependent variable (CVD status)
independent_vars = ['AgeinYr', 'gender', 'height', 'weight', 'systolic BP', 'Diastolic BP', 'cholesterol', 'gluc', 'smoke', 'alco', 'active']
dependent_var = 'cardio'

# Add a constant term to the independent variables
independent_vars = sm.add_constant(df[independent_vars])

# Fit the logistic regression model
logit_model = sm.Logit(df[dependent_var], independent_vars)
logit_result = logit_model.fit()

# Print the summary of the logistic regression model
print(logit_result.summary())

if p > 0.05:
    print('there is no significant relationship between health indicators and cardiovascular disease')
else:
    print('there is a significant relationship between health indicators and cardiovascular disease')
```

Pre-processing the data:

33.
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Split the data into features and target variable
X = df.drop(['id', 'cardio'], axis=1)
y = df['cardio']

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

# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Machine learning models – Logistic regression:

34.
```python
#Logistic Regression:
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# Define features and target variable
X = df[['age', 'gender', 'height', 'weight', 'systolic BP', 'Diastolic BP', 'cholesterol', 'gluc', 'smoke', 'alco', 'active']]
y = df['cardio']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize the model
model = LogisticRegression()

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

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
# Print the performance metrics
print("Logistic Regression Performance Metrics:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

Decision tree model:

35.
```python
#Decision Tree:
from sklearn.tree import DecisionTreeClassifier

# Initialize the model
model = DecisionTreeClassifier()

# Train the model
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the performance metrics
print("Decision Tree Performance Metrics:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

Random Forest model:

36.
```python
#Random Forest:
from sklearn.ensemble import RandomForestClassifier

# Initialize the model
model = RandomForestClassifier()

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

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the performance metrics
print("Random Forest Performance Metrics:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

Support Vector Machine Model:

37.
```python
#Support Vector Machine:
from sklearn.svm import SVC

# Initialize the model
model = SVC()

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

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the performance metrics
print("Support Vector Machine Performance Metrics:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

Data Visualization:

8. DATA VISUALIZATION

38.
```
sns.heatmap(df.isnull(),cbar=False,cmap='viridis')
```

39.
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Define features and target variable
X = df[['age', 'gender', 'height', 'weight', 'systolic BP', 'Diastolic BP', 'cholesterol', 'gluc', 'smoke', 'alco', 'active']]
y = df['cardio']

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

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize the models
models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Support Vector Machine': SVC()
}
```

40.
```
# Train the models and evaluate their performance
results = {'Model': [], 'Accuracy': [], 'Precision': [], 'Recall': [], 'F1 Score': []}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    results['Model'].append(model_name)
    results['Accuracy'].append(accuracy)
    results['Precision'].append(precision)
    results['Recall'].append(recall)
    results['F1 Score'].append(f1)

# Create a table of the results
results_df = pd.DataFrame(results)
print(results_df)
```

41.
```
sns.FacetGrid(df,hue="cardio",height = 5).map(sns.distplot,"BMI"). add_legend()
plt.title('Distribution of Cardiovascular Disease by BMI')
```

42.
```
sns.FacetGrid(df,hue="cardio",height = 5).map(sns.distplot,"AgeinYr"). add_legend()
plt.title('Distribution of Cardiovascular Disease by Age')
```

43.
```
sns.FacetGrid(df,hue="cardio",height = 5).map(sns.distplot,"systolic BP"). add_legend()
plt.title('Distribution of Cardiovascular Disease by systolic BP')
```

44.
```
sns.FacetGrid(df,hue="cardio",height = 5).map(sns.distplot,"Diastolic BP"). add_legend()
plt.title('Distribution of Cardiovascular Disease by diastolic BP')
```

45.
```
import pandas as pd
import matplotlib.pyplot as plt

# Visualize the distribution of a categorical variable
plt.figure(figsize=(10, 6))
df['cholesterol'].value_counts().plot(kind='bar', color='skyblue')
plt.title('Distribution of Cholesterol Levels')
plt.xlabel('Cholesterol Level')
plt.ylabel('Count')
plt.show()
```

46.
```
sns.FacetGrid(df,hue="cardio",height = 5).map(sns.distplot,"cholesterol"). add_legend()
plt.title('Distribution of Cardiovascular Disease by cholesterol levels')
```

47.
```
smoke_vs_cardio = pd.crosstab(index=df["smoke"], columns=df["cardio"])
smoke_vs_cardio.plot(kind="bar", figsize=(8,8),stacked=True)
plt.title('Distribution of Cardiovascular Disease by smoking')
```

48.
```
# Create a cross-tabulation of the "active" and "cardio" variables
active_vs_cardio = pd.crosstab(index=df["active"], columns=df["cardio"])

# Create a bar plot to compare the distribution of the "active" variable by the "cardio" variable
active_vs_cardio.plot(kind="bar", figsize=(8, 8), stacked=True)
plt.title('Distribution of Cardiovascular Disease by Physical Activity')
plt.xlabel('Physical Activity')
plt.ylabel('Count')
plt.show()
```