AIC-202

Programming for Artificial Intelligence

Complex Computing Problem
BS(AI) – 4A

Based on the dataset:
Heart Disease Dataset | Kaggle

Muneeza Iftikhar – 012

Hafsa Hafeez – 026

Haris Aamir – 034

Aqsa Khan – 039

Introduction

In this report, we present a comprehensive analysis of a heart disease dataset, encompassing various factors known to influence cardiovascular health. The dataset includes key variables such as age, angina, cholesterol levels (chol), chest pain type (cp), fasting blood sugar (fbs), maximum heart rate (heartratemax), resting blood pressure (restbps), sex, and the presence of heart disease (targertach). Through rigorous statistical analysis and data exploration, we aim to shed light on the interplay between these variables and heart disease, paving the way for improved risk assessment. [1]

Design And Implementation

```
import tkinter as tk
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from tkinter import ttk
from PIL import ImageTk, Image
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure


# Creating the main window frame
root = tk.Tk()
root.title('Data Analysis')
root.configure(bg='white')
# Designating height and width[2]
def winCentre(window):
    app_Width = 800
    app_Height = 600
    swidth = window.winfo_screenwidth()
    screen_height = window.winfo_screenheight()
    x = (swidth / 2) - (app_Width / 2)
    y = (screen_height / 2) - (app_Height / 2)
    window.geometry(f'{app_Width}x{app_Height}+{int(x)}+{int(y)}')
winCentre(root)
# Inserting the main image[3]
image_path = 'D://BS(AI)//Semester 4//PAI//CCP//many-hearts.jpg'
image = Image.open(image_path)
image = image.resize((400, 300))
image = ImageTk.PhotoImage(image)
image_label = tk.Label(root, image=image)
image_label.pack(pady=20)

main_title = tk.Label(root, text='Data Analysis on Heart Disease',
font=('Roman', 24), bg='white')
main_title.pack(pady=20)
```

[1] (Exercise assessment and measurement of exercise capacity in patients with coronary heart disease)
[2] (How To Center a Tkinter Window On The Screen - Python Tkinter GUI Tutorial #126)
[3] (How To Add Images In Tkinter)

```
def add_dirtydata(file_path):
    # Load the original dataset
    df = pd.read_csv('targetach.csv')
    # Generate random noise for dirty data
    noise = np.random.normal(loc=0, scale=10, size=len(df))
    # Add noise to the selected variables
    df['age_dirty'] = df['age'] + noise
    df['angina_dirty'] = df['angina'] + noise
    df['chol_dirty'] = df['chol'] + noise
    df['heartratemax_dirty'] = df['heartratemax'] + noise
# Display the modified dataset
print(df)

 def input_dataframe(file_path, sort_by):
     # Read dataframe, sort it, and drop null values
     df = pd.read_csv(file_path)
     df = df.sort_values(by=sort_by)
     df = df.dropna()
     return df

 def remove_outliers(df):
     for col in df.columns:
         # Calculate the maximum and minimum values for every column
         max = df[col].mean() + 3 * df[col].std()
         min = df[col].mean() - 3 * df[col].std()

         # Filter the DataFrame on max and min - removing outliers
         df = df[(df[col] > min) & (df[col] < max)]
         return df

 def dataProcessing():
     global targetach  # Make targetach a global variable
     # Data processing
     data_path = "data\\"

     sorting_keys = ['age', 'sex', 'cp']

     targetach = input_dataframe(data_path + "targetach.csv", sorting_keys)
     angina = input_dataframe(data_path + "angina.csv", sorting_keys)
     chol = input_dataframe(data_path + "chol.csv", sorting_keys)
     fbs = input_dataframe(data_path + "fbs.csv", sorting_keys)
     restbps = input_dataframe(data_path + "restbps.csv", sorting_keys)
     heartratemax = input_dataframe(data_path + "heartratemax.csv",
 sorting_keys)

     dataset_list = [angina, chol, fbs, restbps, heartratemax]

     # Treat targetach as main dataset which we will merge on, later &
 indexing column occurs
     targetach = remove_outliers(targetach)
     index = [i + 1 for i in range(int(targetach.size /
 len(targetach.columns)))]
     targetach.insert(0, "index", index)
```

```
    # Iterate through the individual data, clean it, and merge them into the
main dataset
    for item in dataset_list:
        item = remove_outliers(item)
        item = item.drop(columns=sorting_keys, axis=0) # according to column
        index = [i for i in range(int(item.size / len(item.columns)))]
        item.insert(0, "index", index)
        targetach = pd.merge(targetach, item, on=['index'])
    #print(targetach.head(1050))
    return targetach

def openDataSetWindow():
    dataSetWin = tk.Toplevel()
    root.withdraw()
    targetach = dataProcessing()  # Assign the returned DataFrame to
targetach
    # GUI window
    dataSetWin.configure(bg='#FFFFFF')
    label2 = tk.Label(dataSetWin, text="Heart Disease Dataset", bg='light
blue')
    label2.pack(padx=20, pady=20)

    # Create a Text widget to display the dataset
    text_widget = tk.Text(dataSetWin, bg='#FFFFFF')
    text_widget.pack(padx=20, pady=20)

    # Insert the dataset into the Text widget
    text_widget.insert(tk.END, targetach.to_string(index=False))
    text_widget.configure(state='disabled')

    back_button = tk.Button(dataSetWin, text="Back", command=lambda:
backToMain(dataSetWin))
    back_button.pack(side=tk.BOTTOM, pady=20)
    winCentre(dataSetWin)

def visualization1():
    #Query1: No. of people that have angina
    angina_counts = targetach['angina'].value_counts()
    mean_age_angina = targetach.groupby('angina')['age'].mean()
    mean_chol_angina = targetach.groupby('angina')['chol'].mean()
    percentage_angina = (angina_counts[1] / len(targetach)) * 100

    # Create a Tkinter window
    visual1_win = tk.Toplevel()
    visual1_win.title('Data Analysis')
    visual1_win.configure(bg='white')
    winCentre(visual1_win)

    #plotting query 1
    fig = Figure(figsize=(6, 4), dpi=100)      # represents the overall
container for your plot
    ax = fig.add_subplot(111)
    ax.bar(angina_counts.index, angina_counts.values, color='cornflowerblue')
    ax.set_xlabel('Angina (Chest Pain during Exercise)')
    ax.set_ylabel('No. of People')
    ax.set_title('No. of People with Angina')
```

```
    # Create a canvas to embed the plot [4]
    canvas = FigureCanvasTkAgg(fig, master=visual1_win)
    canvas.draw()
    canvas.get_tk_widget().pack()

    meanlabel1 = tk.Label(visual1_win,
                          text=f'Percentage of People experiencing angina
during exercise: {percentage_angina}',
                          font=('Times New Roman', 14),bg = 'white')
    meanlabel1.pack(padx=20, pady=5)

    back_button = tk.Button(visual1_win, text="Back", command=lambda:
backToMain(visual1_win))
    back_button.pack(side=tk.BOTTOM, pady=20)
    visual1_win.mainloop()

def visualization2():
    # query 2
    severity_for_women = targetach[
        (targetach['angina'] == 1) & ((targetach['sex'] == 0) &
(targetach['chol'] > 200)) & (targetach['target'] == 1)]
    severity_for_men = targetach[
        (targetach['angina'] == 1) & ((targetach['sex'] == 1) &
(targetach['chol'] > 200)) & (targetach['target'] == 1)]

    severity_rate_women = (severity_for_women.shape[0] /
                           targetach[(targetach['sex'] == 0) &
(targetach['angina'] == 1)].shape[0]) * 100
    severity_rate_men = (severity_for_men.shape[0] /
                         targetach[(targetach['sex'] == 1) &
(targetach['angina'] == 1)].shape[0]) * 100

    # Plotting query 2
    labels = ['Women', 'Men']
    severity_rates = [severity_rate_women, severity_rate_men]
    colors = ['violet', 'skyblue']
    fig = plt.figure(figsize=(6, 4), dpi=100)
    plt.pie(severity_rates, labels=labels, autopct='%1.1f%%', colors=colors,
startangle=90,
            wedgeprops={'linewidth': 1, 'edgecolor': 'black'})
    plt.axis('equal')
    plt.title('Severity Rate by Gender')

    # Create a Tkinter window
    visual2_win = tk.Toplevel()
    visual2_win.title('Visualization 2')
    visual2_win.configure(bg='white')
    winCentre(visual2_win)

    # Create a canvas to embed the plot[5]
    canvas = FigureCanvasTkAgg(fig, master=visual2_win)
    canvas.draw()
    canvas.get_tk_widget().pack()
```

[4] (Embedding a Matplotlib Graph into a Tkinter application)
[5] (Embedding a Matplotlib Graph into a Tkinter application)

```
    back_button = tk.Button(visual2_win, text="Back", command=lambda:
backToMain(visual2_win))
    back_button.pack(side=tk.BOTTOM, pady=20)

    visual2_win.mainloop()

def visualization3():
    # Map target values to "Yes" and "No"
    targetach['target'] = targetach['target'].replace({0: 'No', 1: 'Yes'})

    # Separate the data for '0' and '1' target values
    target_0 = targetach[targetach['target'] == 'No']
    target_1 = targetach[targetach['target'] == 'Yes']

    # Calculate the mean maximum heart rate achieved for each target value
    mean_hr_0 = target_0['max_heartrate'].mean()
    mean_hr_1 = target_1['max_heartrate'].mean()

    # Plotting query 3
    categories = ['No', 'Yes']
    max_hr_data = [targetach[targetach['target'] == 'No']['max_heartrate'],
                   targetach[targetach['target'] == 'Yes']['max_heartrate']]

    # Print the data used for plotting
    print('Data for category "No":', max_hr_data[0])
    print('Data for category "Yes":', max_hr_data[1])

    fig, ax = plt.subplots(figsize=(6, 4), dpi=100)
    sns.violinplot(data=max_hr_data, inner='quartile', ax=ax)
    ax.set_xticks(ticks=[0, 1])
    ax.set_xticklabels(labels=categories)
    ax.set_xlabel('Heart Disease')
    ax.set_ylabel('Maximum Heart Rate')
    ax.set_title('Distribution of Maximum Heart Rate by Heart Disease')

    # Create a Tkinter window
    visual3_win = tk.Toplevel()
    visual3_win.configure(bg='white')
    visual3_win.title('Visualization 3')
    winCentre(visual3_win)


    # Create a canvas to embed the plot[6]
    canvas = FigureCanvasTkAgg(fig, master=visual3_win)
    canvas.draw()
    canvas.get_tk_widget().pack()

    # Button to close the window
    back_button = tk.Button(visual3_win, text="Back", command=lambda:
backToMain(visual3_win))
    back_button.pack(side=tk.BOTTOM, pady=20)
    visual3_win.mainloop()

def backToMain(window):
    # Show the main window and withdraw the current window
```

[6] (Embedding a Matplotlib Graph into a Tkinter application)

```
    window.withdraw()
    root.deiconify()

def main():
    dataProcessing()
    # Buttons
    # Create a button to open the merged dataset window
    btn2 = tk.Button(root, text="Merged Dataset", command=openDataSetWindow)
    btn2.pack(padx=20, pady=5)

    # Create buttons for each visualization
    btn_visualization1 = tk.Button(root, text="No. Of people that have
Angine", command=visualization1)
    btn_visualization1.pack(padx=20, pady=5)

    btn_visualization2 = tk.Button(root, text="Severity Rate By Gender",
command=visualization2)
    btn_visualization2.pack(padx=20, pady=5)

    btn_visualization3 = tk.Button(root, text="Distribution of Maximum Heart
Rate by Heart Disease", command=visualization3)
    btn_visualization3.pack(padx=20, pady=5)

    # btn_descripitiveStats = tk.Button(root, text = 'Descriptive
Statistics',command=openDescriptiveStatsWin)
    # btn_descripitiveStats.pack(padx=20,pady=5)

    exit_button = tk.Button(root, text="Exit", command=root.destroy)
    exit_button.pack(pady=10)

    root.mainloop()

if_name_== "_main__":
    main()
```

Usage and User Interaction

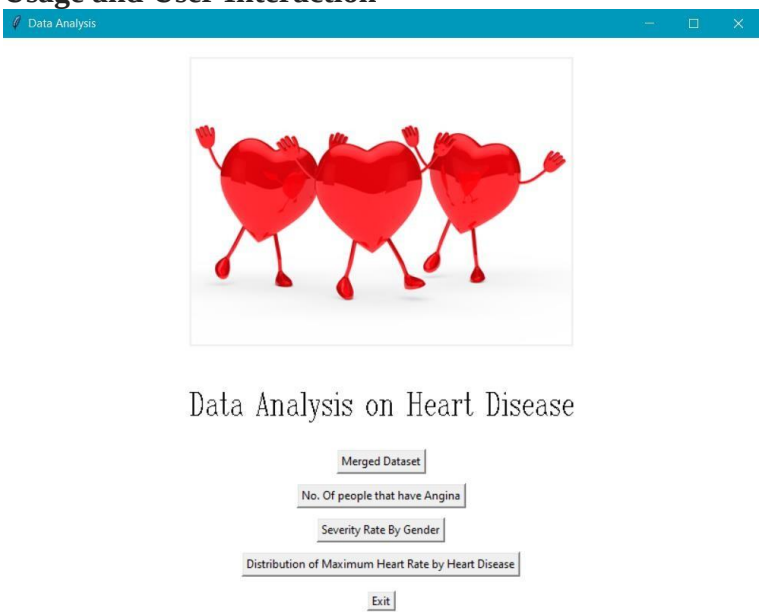


Figure i Data Analysis Main Window[7]

One of the company requirements was to implement user-friendly interactions. This was achieved by making the GUI of the data analysis.[8] It incorporates a main window which includes all of the necessary button for the user to see the visual data based on the columns.

The program analyzes three sets of the heart disease dataset which includes; Number of people that experience angina during exercise, the severity rate of heart attack on the two genders and distribution of the maximum heart rate is experienced by patients with heart disease. The GUI of the program is straightforward and simple. The button of viewing the merged dataset is also accessible to the user if they wish to see it. [9]Since the data was broken and divided into parts the program combines and cleans it for simple and clean view.

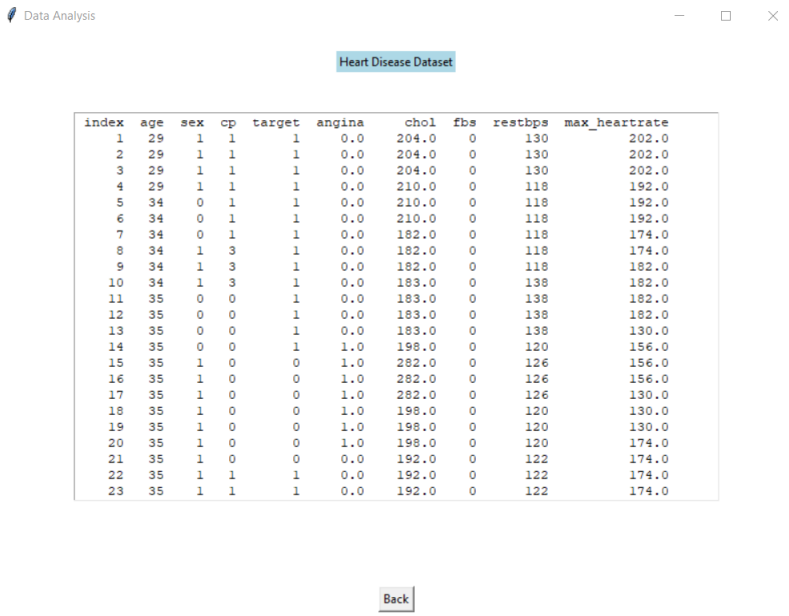
Furthermore, it offers a variety of visualizations to provide users with a comprehensive understanding of the heart disease dataset. Users can choose from several types of visualizations, including bar graphs, pie charts, and violin plots. These visualizations offer different perspectives on the data and allow users to analyze and interpret the information based on their preferences and requirements.

[7] (How To Add Images In Tkinter)
[8] (tkinter — Python interface to Tcl/Tk)
[9] (tkinter — Python interface to Tcl/Tk)

	A	B	C	D	E	F	G	H	I	J
1	age	sex	cp	chol						
2	52	1	0	212						
3	53	1	0	203						
4	70	1	0	174						
5	61	1	0	203						
6	62	0	0	294						
7	58	0	0	248						
8	58	1	0	318						
9	55	1	0	289						
10	46	1	0							
11	54	1	0	286						
12	71	0	0	149						
13	43	0	0	341						
14	34	0	1	210						
15	51	1	0	298						
16	52	1	0	204						
17	34	0	1	210						
18	51	0	2	308						
19	54	1	0	266						
20	50	0	1	7889						
21	58	1	2	211						

*Figure ii One of the *.csv files to show the existing the null values and outliers that lie in the dataset.*

Adding dirty data helps evaluate the effectiveness of data quality assurance processes and algorithms. It allows us to test the efficiency of data cleaning and outlier detection methods in identifying and handling erroneous or inconsistent data. By intentionally introducing known issues, we can assess the accuracy and reliability of data cleaning techniques.



index	age	sex	cp	target	angina	chol	fbs	restbps	max_heartrate
1	29	1	1	1	0.0	204.0	0	130	202.0
2	29	1	1	1	0.0	204.0	0	130	202.0
3	29	1	1	1	0.0	204.0	0	130	202.0
4	29	1	1	1	0.0	210.0	0	118	192.0
5	34	0	1	1	0.0	210.0	0	118	192.0
6	34	0	1	1	0.0	210.0	0	118	192.0
7	34	0	1	1	0.0	182.0	0	118	174.0
8	34	1	3	1	0.0	182.0	0	118	174.0
9	34	1	3	1	0.0	182.0	0	118	182.0
10	34	1	3	1	0.0	183.0	0	138	182.0
11	35	0	0	1	0.0	183.0	0	138	182.0
12	35	0	0	1	0.0	183.0	0	138	182.0
13	35	0	0	1	0.0	183.0	0	138	130.0
14	35	0	0	1	1.0	198.0	0	120	156.0
15	35	1	0	0	1.0	282.0	0	126	156.0
16	35	1	0	0	1.0	282.0	0	126	156.0
17	35	1	0	0	1.0	282.0	0	126	130.0
18	35	1	0	0	1.0	198.0	0	120	130.0
19	35	1	0	0	1.0	198.0	0	120	130.0
20	35	1	0	0	1.0	198.0	0	120	174.0
21	35	1	0	0	0.0	192.0	0	122	174.0
22	35	1	1	1	0.0	192.0	0	122	174.0
23	35	1	1	1	0.0	192.0	0	122	174.0

Figure iii The dataset was broken and divided into separate columns. Here, this window displays the dataset cleaned and merged together for the user to view.

Given the importance of clean data in a data centric system, preprocessing and cleaning are essential steps in preparing the dataset. This dataset was cleaned by removing outliers and dropping null values which are both common preprocessing techniques to handle disproportionate and missing data, respectively.

All the *.csv files were cleaned and preprocessed as required and merged into a single dataset to be displayed to the user.

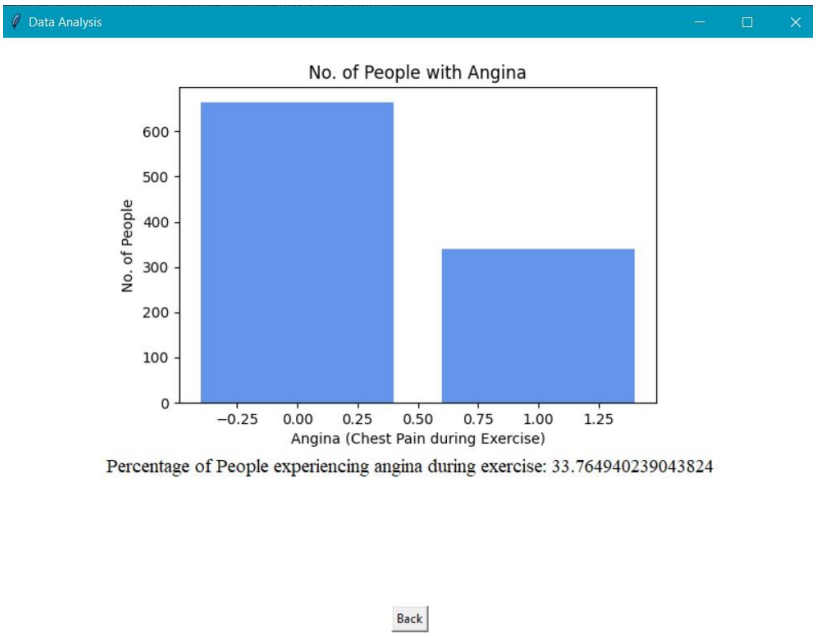


Figure iv This window displays query 1 which analyzes number of people who experience angina during exercise. The data is displayed through a bar graph and displays the accumulated percentage which is calculated from the merged dataset. The x-axis represents angina and y-axis represents no. of people [10]

The purpose of this query is to analyze the presence of angina (chest pain during exercise) in the dataset and calculate relevant statistics.

Angina is an important symptom related to heart disease and can provide insights into the severity and impact of cardiovascular conditions.[11] By examining the occurrence of angina in the dataset, we can assess its prevalence among the participants. Counting the occurrences of angina allows us to determine the number of individuals who experience chest pain during exercise. This information is useful in understanding the overall prevalence of angina within the dataset.[12]

Additionally, calculating the average age and average cholesterol for participants with angina helps us understand the characteristics of individuals experiencing angina. Age and cholesterol levels are known risk factors for heart disease, and analyzing their values specifically for participants with angina provides insights into the relationship between angina and these risk factors. By calculating the mean age and mean cholesterol for participants with angina, we can identify any potential patterns or associations between these variables and the occurrence of angina.[13]

Furthermore, determining the percentage of participants with angina helps us quantify the proportion of individuals experiencing chest pain during exercise relative to the entire dataset. This percentage provides a measure of the overall impact of angina within the population under study.

[10] (Pro, 2015)
[11] (Angina) (Angina Pectoris)
[12] (Exercise assessment and measurement of exercise capacity in patients with coronary heart disease)
[13] (Heart Disease and Stroke Statistics—2021 Update)

Choosing a bar chart for this query is justified for the following reasons[14]:

Categorical Variable: In this query, we are analyzing the presence or absence of angina (chest pain) during exercise, which is a categorical variable. Bar charts are well-suited for displaying and comparing categorical data, making them a suitable choice for visualizing the counts or frequencies of different categories.

Count Comparison: Bar charts allow you to easily compare the counts of different categories. In our case, we used a bar chart to show the number of people who have angina and those who do not. The length of the bars represents the count, making it easy to identify any differences or patterns between the two categories.

Simple and Clear Representation: Bar charts provide a clear and straightforward representation of categorical data. Each category is represented by a separate bar, and the height of the bar corresponds to the count or frequency. This simplicity makes it easy to interpret and understand the information being presented.

Emphasizing Individual Categories: Bar charts can effectively emphasize individual categories and highlight their respective counts. By using different colors or patterns for the bars representing "No Angina" and "Angina," you can further enhance the visual distinction between the two categories and draw attention to their respective counts.

Facilitating Comparison and Analysis: Bar charts allow for easy visual comparison and analysis of categorical data. You can quickly identify which category has a higher count and assess the relative proportions between the two categories.

Choosing a bar chart for this query is justified as it enables a clear and concise representation of the counts or frequencies of angina presence and absence during exercise. It allows for easy comparison between the categories and provides a visual means to analyze the distribution of the variable.

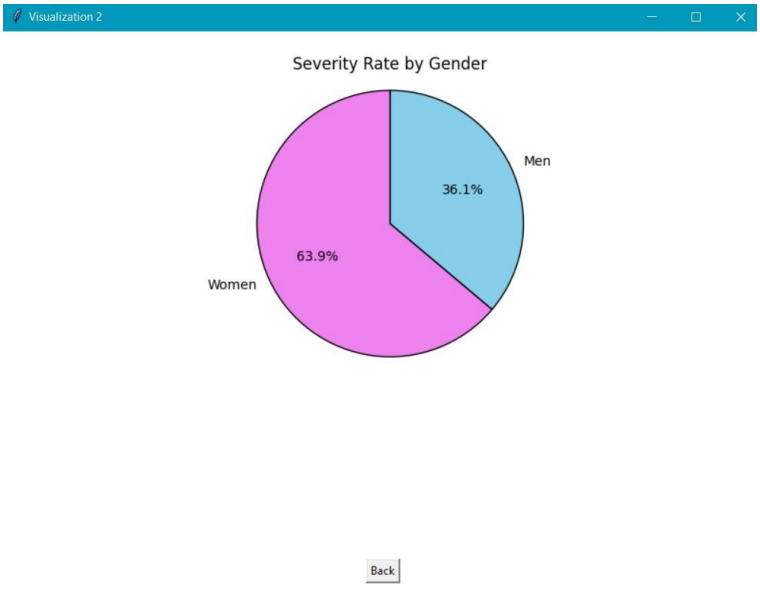


Figure v This is the second visualization which analyzes the severity rate of heart attack experienced by men and women which is

[14] (Yi, A Complete Guide to Stacked Bar Charts, n.d.)

represented through a percentage for a more concise understanding through analyzing cholesterol and angina.[15]

This query aims to examine the severity rate of heart disease among women and men who experience angina and have high cholesterol.

Understanding the severity rate of heart disease in different population segments provides valuable insights into the impact and characteristics of the condition. In this query, we focus specifically on individuals who experience angina and have high cholesterol levels, as these factors are indicative of potential cardiovascular issues.

By filtering the dataset based on specific conditions related to angina, sex, cholesterol, and the presence of heart disease, we can identify subsets of the data that meet these criteria. Separating the data for women and men and applying additional conditions allows us to isolate individuals with angina and high cholesterol within each gender category.

The calculation of severity rates for women and men separately provides a measure of the proportion of individuals with heart disease among those who experience angina and have high cholesterol. By comparing the severity rates between genders, we can identify any potential gender-based differences in the impact and severity of heart disease in the context of angina and high cholesterol.

Analyzing the severity rates in this manner helps to assess the relative risk and severity of heart disease in specific subgroups. It provides valuable information for understanding the gender-specific implications of angina and high cholesterol on the development and severity of heart disease.

Choosing a pie chart is justified for the following reasons[16]:

Comparison of Parts to a Whole: A pie chart is well-suited for representing parts of a whole. In this case, comparing the severity rates for two groups (males and females) as parts of the overall population. The pie chart can visually show the proportion of severity rates for each group in relation to the total.
Emphasizing Proportions and Percentages: Pie charts excel in highlighting proportions and percentages. By displaying the severity rates for males and females as slices of a pie, it becomes easy to see the relative sizes of the rates and compare them visually. The chart can also include percentage labels on each slice, providing clear information about the proportion of severity rates for each group.

Simplicity and Clarity: Pie charts are simple and intuitive, making them easily interpretable for a wide range of audiences. They offer a concise and clear representation of the data, allowing viewers to quickly grasp the relative severity rates for males and females without being overwhelmed by excessive details or axes.

Limited Number of Categories: Pie charts are most effective when representing a small number of categories. In this query, we are comparing severity rates for only two categories (males and females). Pie charts are particularly suitable for such scenarios where there are a limited number of groups to compare.
Since the primary goal is to emphasize the comparison of severity rates between males and females as proportions of the whole, a pie chart can be a justifiable choice to visually convey this information effectively.

[15] (Undatanble, 2021)
[16] (Yi, How to Choose Between a Bar Chart and Pie Chart, n.d.)

Figure vi The following data is represented through a violin plot. The plot represents the maximum heartrate between individuals with the disease and the individuals without it. The x-axis represents whether the person has the disease or not and the y axis represents the maximum heart rate achieved during exercise.

This query focuses on exploring the distribution of the maximum heart rate achieved based on the presence or absence of heart disease.

The maximum heart rate achieved during exercise is an essential measure of cardiovascular fitness and can provide insights into the physiological response of the heart. Analyzing the distribution of maximum heart rate in relation to the presence or absence of heart disease allows us to assess the potential impact of cardiovascular conditions on exercise capacity.[17]

Separating the data into two groups based on heart disease status allows us to compare the distribution of maximum heart rate achieved between individuals with and without heart disease. By calculating the mean maximum heart rate achieved for each group, we can evaluate any differences in the average maximum heart rate based on the presence or absence of heart disease.

In this query, the goal of calculating the average maximum heart rate for both individuals with heart disease and individuals without heart disease is to understand the relationship between heart disease and the maximum heart rate achieved during exercise.

The maximum heart rate achieved during exercise is an indicator of cardiovascular fitness. Generally, individuals with higher fitness levels tend to achieve higher maximum heart rates during physical activity. By examining the average maximum heart rate for individuals with and without heart disease, we can gain insights into the impact of heart disease on cardiovascular fitness. It helps us understand whether individuals with heart disease lower maximum heart rates on average, indicating compromised cardiovascular function.

Choosing a violin plot is justified for the following reasons[18]:

[17] (Angina)
[18] (Yi, A Complete Guide to Violin Plots, n.d.)

Distribution Comparison: Violin plots are particularly useful when comparing the distributions of a variable across different categories or groups. In query exploring the distribution of maximum heart rate achieved during exercise between individuals with and without heart disease. A violin plot allows you to visualize and compare the shape, spread, and density of the distributions for each group.

Continuous Variable: Violin plots are well-suited for representing continuous variables, such as the maximum heart rate achieved. They provide insights into the distribution characteristics, including the median, quartiles, and potential outliers. This can be valuable for understanding the range and central tendencies of the variable in relation to the presence or absence of heart disease.

Visualizing Multiple Groups: Violin plots can effectively handle multiple groups and display their distributions side by side. In query, we have two groups: individuals with no heart disease and individuals with heart disease. By using a violin plot, you can easily compare the distribution of maximum heart rate achieved for each group, observe any differences or patterns, and identify potential relationships with cardiovascular fitness.

Capturing Summary Statistics: Violin plots provide a visual representation of summary statistics, such as the median, quartiles, and density estimation. These elements offer a concise summary of the distribution characteristics for each group, enabling quick insights into the differences or similarities between individuals with and without heart disease.

Conclusion

This report underscores the significance of examining the relationships between heart disease and associated variables. The analysis of age, angina, cholesterol levels, chest pain type, fasting blood sugar, heart rate during exercise, resting blood pressure, sex, and presence of heart disease (targetach) reveals significant associations and provides valuable insights into the complex nature of heart disease.

References

Angina. (n.d.). Retrieved from Mayo Clinic: https://www.mayoclinic.org/diseases-conditions/angina/symptoms-causes/syc-20369373

Angina Pectoris. (n.d.). Retrieved from MSD MANUAL: https://www.msdmanuals.com/professional/cardiovascular-disorders/coronary-artery-disease/angina-pectoris

Embedding a Matplotlib Graph into a Tkinter application. (n.d.). Retrieved from Youtube: https://www.youtube.com/watch?v=lVTC8CvScQo

Exercise assessment and measurement of exercise capacity in patients with coronary heart disease. (n.d.). Retrieved from UpToDate: https://www.uptodate.com/contents/exercise-assessment-and-measurement-of-exercise-capacity-in-patients-with-coronary-heart-disease

Heart Disease and Stroke Statistics—2021 Update. (n.d.). Retrieved from Circulation: https://www.ahajournals.org/doi/10.1161/CIR.0000000000000950

How To Add Images In Tkinter. (n.d.). Retrieved from ActiveState: https://www.activestate.com/resources/quick-reads/how-to-add-images-in-tkinter/

How To Center a Tkinter Window On The Screen - Python Tkinter GUI Tutorial #126. (n.d.). Retrieved from Youtube: https://www.youtube.com/watch?v=TdTks2eSx3c

Pro, S. L. (2015, Oct 17). *Bar Charts, Pie Charts, Histograms, Stemplots, Timeplots (1.2)*. Retrieved from Youtube: https://www.youtube.com/watch?v=uHRqkGXX55I&list=TLPQMDQwNjIwMjPQxx9BZ1ggnA&index=2&ab_channel=SimpleLearningPro

Target Heart Rates Chart. (n.d.). Retrieved from Heart Attack and Stroke Symptoms: https://www.heart.org/en/healthy-living/fitness/fitness-basics/target-heart-rates

tkinter — Python interface to Tcl/Tk. (n.d.). Retrieved from Python: https://docs.python.org/3/library/tkinter.html#important-tk-concepts

Undatanble. (2021, May 27). *How To Choose The Right Graph (Types of Graphs and When To Use Them)*. Retrieved from Youtube: https://www.youtube.com/watch?v=o7F-tbBl_hA&list=TLPQMDQwNjIwMjPQxx9BZ1ggnA&index=3&ab_channel=UNDATABLE

Yi, M. (n.d.). *A Complete Guide to Stacked Bar Charts*. Retrieved from Chartio: https://chartio.com/learn/charts/stacked-bar-chart-complete-guide/

Yi, M. (n.d.). *A Complete Guide to Violin Plots*. Retrieved from Chartio: https://chartio.com/learn/charts/violin-plot-complete-guide/

Yi, M. (n.d.). *How to Choose Between a Bar Chart and Pie Chart*. Retrieved from Chartio: https://chartio.com/learn/charts/how-to-choose-pie-chart-vs-bar-chart/