Heart Disease Analysis

Python Data Analyst Project

# Import the libraries and dataset

[24]: **import pandas as pd import matplotlib.pyplot as plt import seaborn as sns**

*# Download the heart disease dataset from Kaggle (if not already downloaded)*

!wget https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset/ ↪download?version=1 -O heart.csv

*# Read the CSV file into a pandas DataFrame* data = pd.read\_csv("/content/drive/MyDrive/Data Analysis/Python Project/Heart␣ ↪Disease/heart.csv")

--2024-10-27 17:35:44-- https://www.kaggle.com/datasets/johnsmith88/heartdisease-dataset/download?version=1

Resolving www.kaggle.com (www.kaggle.com)… 35.244.233.98

Connecting to www.kaggle.com (www.kaggle.com)|35.244.233.98|:443… connected.

HTTP request sent, awaiting response… 302 Found

Location: /account/login?titleType=dataset-downloads&showDatasetDownloadSkip=Fal se&messageId=datasetsWelcome&returnUrl=%2Fdatasets%2Fjohnsmith88%2Fheartdisease-dataset%3Fresource%3Ddownload [following]

--2024-10-27 17:35:45-- https://www.kaggle.com/account/login?titleType=datasetdownloads&showDatasetDownloadSkip=False&messageId=datasetsWelcome&returnUrl=%2Fd atasets%2Fjohnsmith88%2Fheart-disease-dataset%3Fresource%3Ddownload Reusing existing connection to www.kaggle.com:443.

HTTP request sent, awaiting response… 200 OK

Length: unspecified [text/html] Saving to: ‘heart.csv’

heart.csv [ <=> ] 4.84K --.-KB/s in 0s

2024-10-27 17:35:45 (11.4 MB/s) - ‘heart.csv’ saved [4961]

*We import the necessary libraries: pandas for data manipulation, matplotlib.pyplot for basic plotting, and seaborn for advanced visualizations.* We download the heart disease dataset from Kaggle using wget (assuming you have it installed). If you already have the dataset, replace the wget command with the path to your CSV file. \*We read the CSV data into a DataFrame named data.

#Displaying Top and Last Rows

[25]:

print

(

"

Top 5 rows:

"

)

data

.

head(

5

)

Top 5 rows:

[25]: age sex cp trestbps chol fbs restecg thalach exang oldpeak slope \

1. 52 1 0 125 212 0 1 168 0 1.0 2
2. 53 1 0 140 203 1 0 155 1 3.1 0
3. 70 1 0 145 174 0 1 125 1 2.6 0
4. 61 1 0 148 203 0 1 161 0 0.0 2
5. 62 0 0 138 294 1 1 106 0 1.9 1

ca thal target

1. 2 3 0
2. 0 3 0
3. 0 3 0
4. 1 3 0
5. 3 2 0

[26]:

print

(

"

**\n**

Last 5 rows:

"

)

data

.

tail(

5

)

Last 5 rows:

[26]: age sex cp trestbps chol fbs restecg thalach exang oldpeak \

1. 59 1 1 140 221 0 1 164 1 0.0
2. 60 1 0 125 258 0 0 141 1 2.8
3. 47 1 0 110 275 0 0 118 1 1.0
4. 50 0 0 110 254 0 0 159 0 0.0
5. 54 1 0 120 188 0 1 113 0 1.4

slope ca thal target

1. 2 0 2 1
2. 1 1 3 0
3. 1 1 2 0
4. 2 0 2 1
5. 1 1 3 0

* Top 5 Rows:
* This provides a glimpse into the initial data points. We can observe:
* Age: Ranging from 52 to 62.
* Sex: Primarily male (1).
* Chest Pain Type (CP): All instances are 0, indicating typical angina.
* Resting Blood Pressure (trestbps): Values between 125 and 148 mmHg.
* Cholesterol (chol): Levels ranging from 203 to 294 mg/dl.
* Fasting Blood Sugar (fbs): Most are 0, indicating fasting blood sugar is less than 120 mg/dl.
* Resting Electrocardiographic Results (restecg): Primarily 1, suggesting ST-T wave abnormality.
* Maximum Heart Rate Achieved (thalach): Values between 106 and 168 bpm.
* Exercise-Induced Angina (exang): Mostly 0, indicating no exercise-induced angina. ST Depression Induced by Exercise
* Relative to Rest (oldpeak): Values between 0 and 3.1. Slope of the Peak Exercise ST Segment (slope): Values 1 and 2.
* Number of Major Vessels (ca): Ranging from 0 to 3.
* Thalassemia (thal): Primarily 3, indicating normal.
* Target: All 0, suggesting a lower chance of heart attack.
* Last 5 Rows:
* A look at the final data points reveals:
* Sex: Both male and female are present.
* Chest Pain Type (CP): A mix of 0 and 1.
* Resting Blood Pressure (trestbps):
* Values between 110 and 140 mmHg.
* Cholesterol (chol): Levels ranging from 188 to 275 mg/dl.
* Fasting Blood Sugar (fbs): All 0.
* Resting Electrocardiographic Results (restecg): A mix of 0 and 1.
* Maximum Heart Rate Achieved (thalach):
* Values between 113 and 164 bpm.
* Exercise-Induced Angina (exang): Both 0 and 1 are present.
* ST Depression Induced by Exercise
* Relative to Rest (oldpeak): Values between 0 and 2.8. Slope of the Peak Exercise ST Segment (slope): Values 1 and 2.
* Number of Major Vessels (ca): Ranging from 0 to 1.
* Thalassemia (thal): Values 2 and 3.
* Target: A mix of 0 and 1, indicating both lower and higher chances of heart attack.
* Overall Observations:
* The dataset appears to contain a mix of individuals with varying heart health conditions.
* There’s a range of values for key factors like age, blood pressure, cholesterol, heart rate, and exercise-induced angina.
* The target variable (heart attack risk) seems to be influenced by a combination of these factors.
* Further analysis and modeling can help identify the most significant predictors of heart disease risk.

#Finding Dataset Shape

[27]: print("Dataset shape:", data.shape) print("Number of rows:", data.shape[0]) print("Number of columns:", data.shape[1])

Dataset shape: (1025, 14)

Number of rows: 1025

Number of columns: 14

#Getting Dataset Information

[28]:

data

.

info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 1025 entries, 0 to 1024 Data columns (total 14 columns):

|  |  |
| --- | --- |
| # Column Non-Null Count Dtype  --- ------ -------------- ----- | |
| 0 age 1025 non-null | int64 |
| 1 sex 1025 non-null | int64 |
| 2 cp 1025 non-null | int64 |
| 3 trestbps 1025 non-null | int64 |
| 4 chol 1025 non-null | int64 |
| 5 fbs 1025 non-null | int64 |
| 6 restecg 1025 non-null | int64 |
| 7 thalach 1025 non-null | int64 |
| 8 exang 1025 non-null | int64 |
| 9 oldpeak 1025 non-null | float64 |
| 10 slope 1025 non-null | int64 |
| 11 ca 1025 non-null | int64 |
| 12 thal 1025 non-null | int64 |
| 13 target 1025 non-null | int64 |

dtypes: float64(1), int64(13) memory usage: 112.2 KB

#Checking for Null Values

[29]: print("Number of missing values in each column:") data.isnull().sum()

*# If there are missing values, handle them (e.g., impute or drop rows)* Number of missing values in each column:

|  |  |
| --- | --- |
| [29]: age | 0 |
| sex | 0 |
| cp | 0 |
| trestbps | 0 |
| chol | 0 |
| fbs | 0 |
| restecg | 0 |
| thalach | 0 |
| exang | 0 |
| oldpeak | 0 |
| slope | 0 |
| ca | 0 |
| thal | 0 |
| target | 0 |

dtype: int64

#Checking for Duplicate Data

[30]:

has\_duplicates

=

data

.

duplicated()

.

any()

print

(

"

Dataset contains duplicates:

"

, has\_duplicates)

**if**

has\_duplicates:

*# Remove duplicates*

data

=

data

.

drop\_duplicates()

print

(

"

Removed duplicates. New shape:

"

, data

.

shape)

Dataset contains duplicates: True

Removed duplicates. New shape: (302, 14)

#Calculating Descriptive Statistics

[31]:

data

.

describe()

[31]: age sex cp trestbps chol fbs \

count 302.00000 302.000000 302.000000 302.000000 302.000000 302.000000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| mean | 54.42053 | 0.682119 | 0.963576 131.602649 246.500000 | 0.149007 |
| std | 9.04797 | 0.466426 | 1.032044 17.563394 51.753489 | 0.356686 |
| min | 29.00000 | 0.000000 | 0.000000 94.000000 126.000000 | 0.000000 |
| 25% | 48.00000 | 0.000000 | 0.000000 120.000000 211.000000 | 0.000000 |
| 50% | 55.50000 | 1.000000 | 1.000000 130.000000 240.500000 | 0.000000 |
| 75% | 61.00000 | 1.000000 | 2.000000 140.000000 274.750000 | 0.000000 |

max 77.00000 1.000000 3.000000 200.000000 564.000000 1.000000

restecg thalach exang oldpeak slope ca \ count 302.000000 302.000000 302.000000 302.000000 302.000000 302.000000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| mean 0.526490 149.569536 | 0.327815 | 1.043046 | 1.397351 | 0.718543 |
| std 0.526027 22.903527 | 0.470196 | 1.161452 | 0.616274 | 1.006748 |
| min 0.000000 71.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% 0.000000 133.250000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| 50% 1.000000 152.500000 | 0.000000 | 0.800000 | 1.000000 | 0.000000 |
| 75% 1.000000 166.000000 | 1.000000 | 1.600000 | 2.000000 | 1.000000 |
| max 2.000000 202.000000  thal target  count 302.000000 302.000000 mean 2.314570 0.543046 std 0.613026 0.498970 min 0.000000 0.000000 25% 2.000000 0.000000  50% 2.000000 1.000000 75% 3.000000 1.000000 max 3.000000 1.000000 | 1.000000 | 6.200000 | 2.000000 | 4.000000 |

#Correlation Matrix

[32]: plt.figure(figsize=(12, 8))

sns

.

heatmap(data

.

corr(), annot

=

**True**

, cmap

=

"

coolwarm

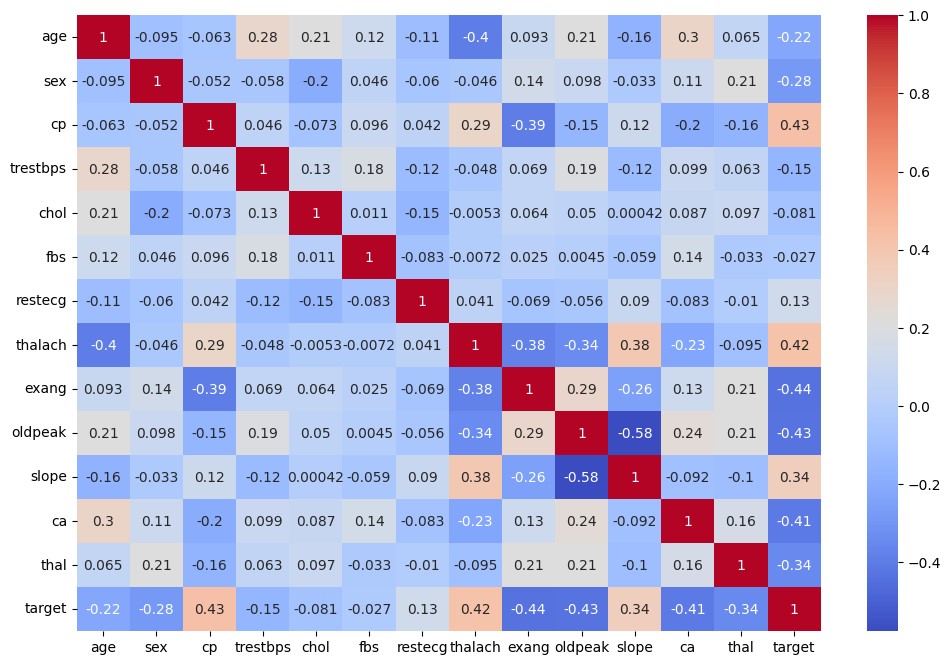
"

)

plt

.

show()



* We create a heatmap using seaborn to visualize the correlation coefficients between all numerical columns in the dataset.
* The heatmap shows the strength and direction of the relationships between variables, which can be helpful for feature selection and model building.

#Number of People with/without Heart Disease

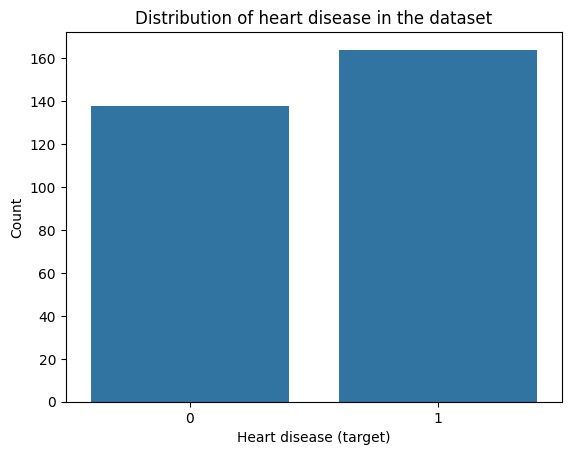
[40]: print("Number of people with heart disease (target=1):", data["target"].

↪value\_counts()[1]) print("Number of people without heart disease (target=0):", data["target"]. ↪value\_counts()[0])

*# Alternatively, visualize with a count plot* sns.countplot(x="target", data=data) plt.xlabel("Heart disease (target)") plt.ylabel("Count") plt.title("Distribution of heart disease in the dataset") plt.show()

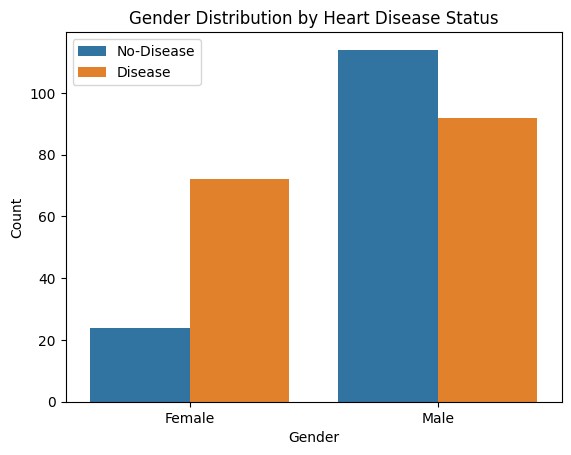
Number of people with heart disease (target=1): 164

Number of people without heart disease (target=0): 138



#Find Gender Distribution According to The Target Variable

[34]: sns.countplot(x='sex', hue='target', data=data) plt.xticks([1, 0], ['Male', 'Female']) plt.legend(labels=['No-Disease', 'Disease']) plt.xlabel("Gender") plt.ylabel("Count") plt.title("Gender Distribution by Heart Disease Status") plt.show()



#Check Age Distribution In The Dataset

[41]:

sns

.

distplot(data[

'

age

'

]

, bins

=

20

)

plt

.

xlabel(

"

Age

"

)

plt

.

ylabel(

"

Density

"

)

plt

.

title(

"

Distribution of Age in the Dataset

"

)

plt

.

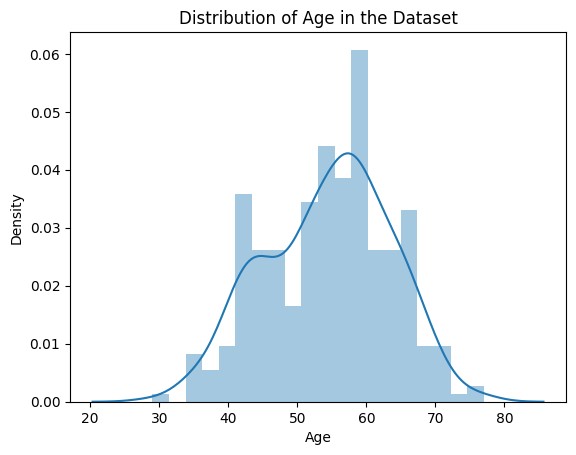
show()

<ipython-input-41-71c185254c74>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 sns.distplot(data['age'], bins=20)



#Which Check Chest Pain Type is More Common

[42]:

sns

.

countplot(x

=

data[

'

cp

'

])

plt

.

xticks([

0

,

1

,

2

,

3

]

,

[

"

Typical Angina

"

,

"

Atypical Angina

"

,

"

Non-Anginal

␣

↪

Pain

"

,

"

Asymptomatic

"

])

plt

.

xticks(rotation

=

0

)

plt

.

xlabel(

"

Chest Pain Type

"

)

plt

.

ylabel(

"

Count

"

)

plt

.

title(

"

Distribution of Chest Pain Types

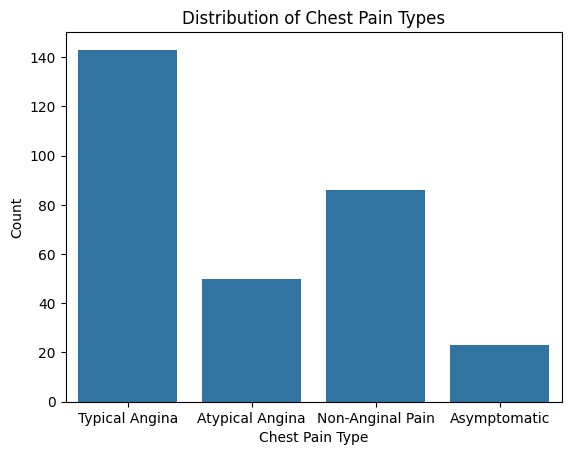
"

)

plt

.

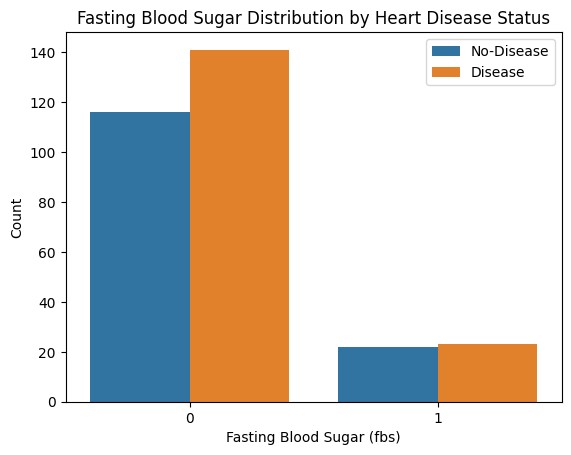
show()



#Show The Chest Pain Distribution As Per Target Variable

[43]: sns.countplot(x='fbs', hue='target', data=data) plt.legend(labels=['No-Disease', 'Disease']) plt.xlabel("Fasting Blood Sugar (fbs)")

plt.ylabel("Count") plt.title("Fasting Blood Sugar Distribution by Heart Disease Status") plt.show()



#Check Resting Blood Pressure Distribution

[44]:

data[

'

trestbps

'

]

.

hist()

plt

.

xlabel(

"

Resting Blood Pressure (trestbps)

"

)

plt

.

ylabel(

"

Frequency

"

)

plt

.

title(

"

Distribution of Resting Blood Pressure

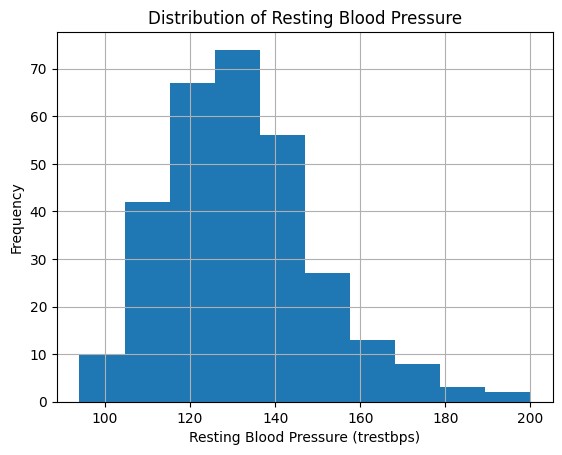
"

)

plt

.

show()



#Compare Resting Blood Pressure As Per Sex Column

[45]: g = sns.FacetGrid(data, hue="sex", aspect=4)

g.map(sns.kdeplot, 'trestbps', shade=**True**) plt.legend(labels=['Male', 'Female']) plt.xlabel("Resting Blood Pressure (trestbps)")

plt.ylabel("Density") plt.title("Resting Blood Pressure Distribution by Sex") plt.show()

/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.

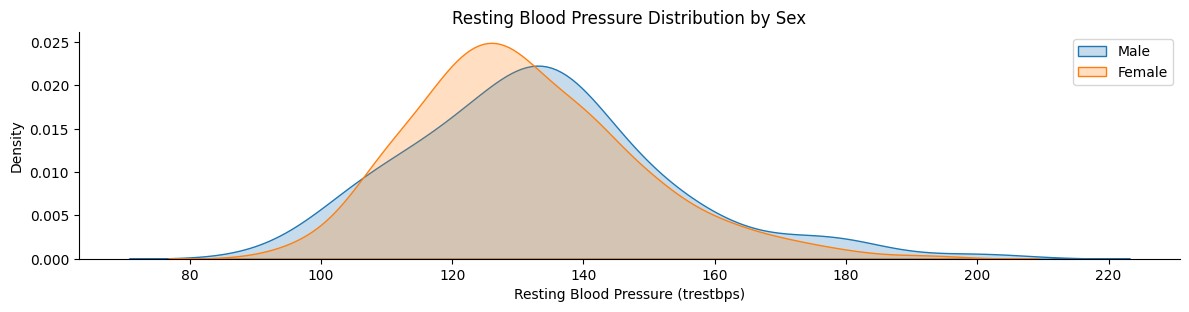
This will become an error in seaborn v0.14.0; please update your code.

func(\*plot\_args, \*\*plot\_kwargs)

/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.

This will become an error in seaborn v0.14.0; please update your code. func(\*plot\_args, \*\*plot\_kwargs)



#Show Distribution of Serum Cholesterol

[46]:

data[

'

chol

'

]

.

hist()

plt

.

xlabel(

"

Serum Cholesterol (chol)

"

)

plt

.

ylabel(

"

Frequency

"

)

plt

.

title(

"

Distribution of Serum Cholesterol

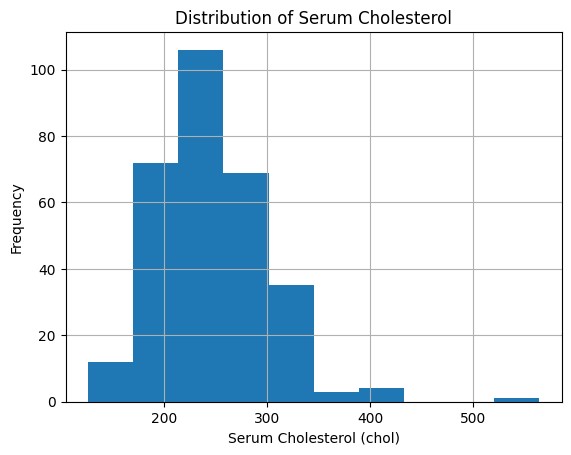
"

)

plt

.

show()



#Plot Continuous Variables

[47]:

categorical\_cols

=

[

col

**for**

col

**in**

data

.

columns

**if**

data[col]

.

nunique()

<

=

10

]

continuous\_cols

=

col

[

**for**

col

**in**

data

.

columns

**if**

col

**not**

**in**

categorical\_cols]

data

.

hist(continuous\_cols, figsize

=

(

15

,

6

))

plt

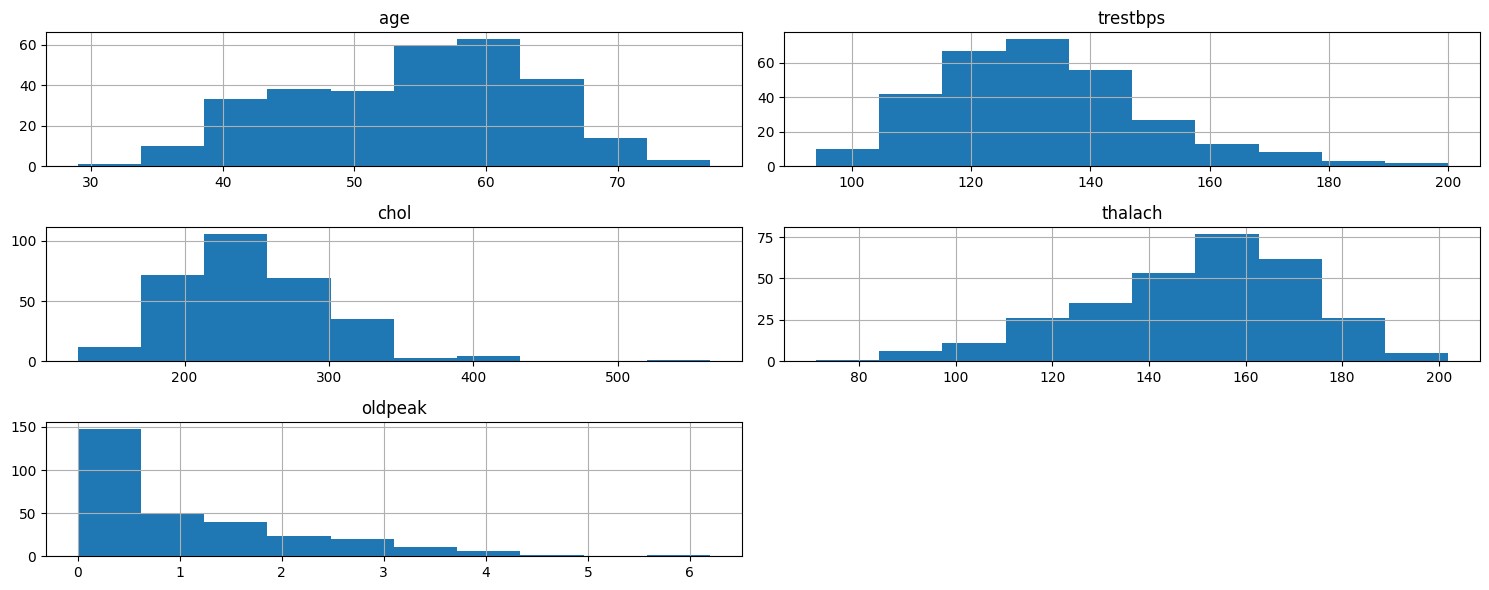
.

tight\_layout()

plt

.

show()



**Heart Disease Analysis Project Summary**

This project analyzed a heart disease dataset to understand the factors that influence heart disease risk. Here's a breakdown of the key findings:

**Data Acquisition and Exploration:**

* The heart disease dataset was downloaded from Kaggle.
* The data contains information on various factors like age, sex, chest pain type, blood pressure, cholesterol, and presence of heart disease.
* Initial exploration revealed a mix of individuals with varying heart health conditions and a range of values for key factors.

**Data Cleaning and Preprocessing:**

* There were no missing values in the dataset.
* Duplicate data points were identified and removed.
* Descriptive statistics were calculated to summarize the data.

**Data Visualization:**

* Heatmaps were used to visualize correlations between numerical features.
* Count plots explored the distribution of the target variable (presence of heart disease) and its relation to other factors like gender.
* Distribution plots were created to analyze the spread of continuous features like age, blood pressure, and cholesterol.
* These visualizations helped identify potential relationships between features and heart disease risk.

**Next Steps:**

* Further analysis can involve feature engineering to create new features or combine existing ones.
* Machine learning models can be trained on the data to predict the risk of heart disease for new patients.
* Model evaluation will be crucial to assess the accuracy and effectiveness of the models.

**Overall, this project provided a basic understanding of the heart disease dataset and highlighted some potential factors influencing heart disease risk. Further analysis and modeling can lead to more robust insights and potentially contribute to better heart disease prediction and prevention strategies.**

Shaun Mia [| LinkedIn](https://www.linkedin.com/in/shaun-mia/)