PREDICTING HEART DISEASE

Data Analytics Assignment Report

Problem Statement

Heart disease remains a significant global health concern, affecting millions of lives. The critical question arises: "How can we predict heart disease before it occurs?" Through supervised machine learning, this project aims to develop a predictive model to detect heart disease, using a target variable of 1 (heart disease detected) and 0 (no heart disease).

To address this issue, a balanced dataset from Kaggle is analysed, comprising 303 individuals—165 with heart disease (target = 1) and 138 without (target = 0). The dataset includes various features, such as age, sex, cholesterol levels, resting heart rates, and more, capturing a comprehensive range of risk factors.

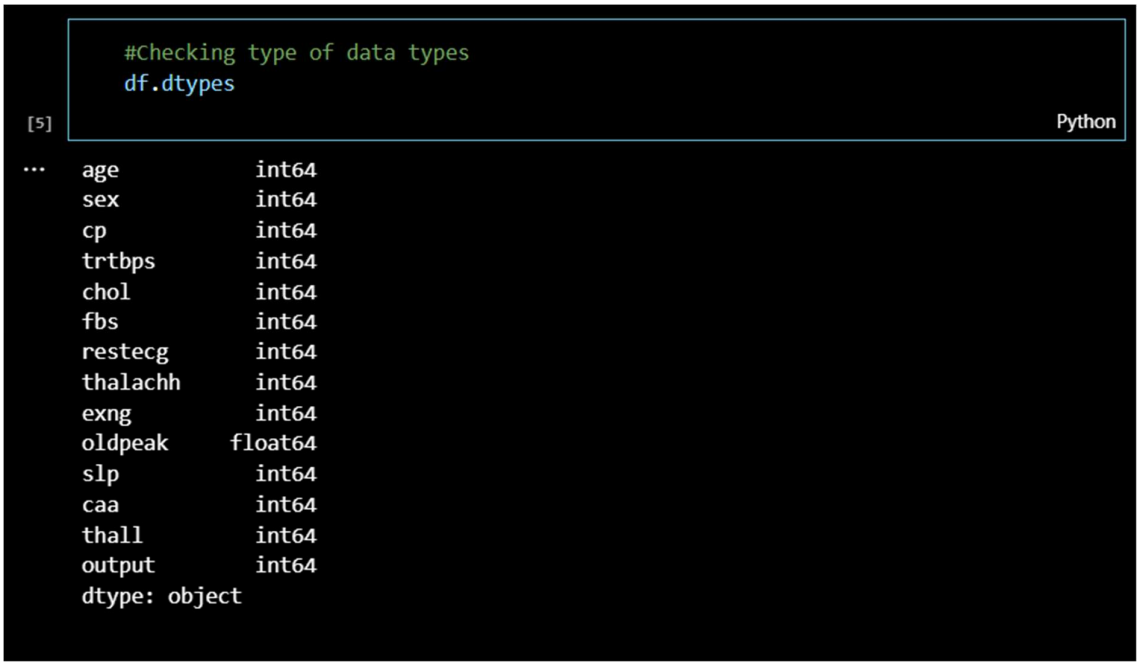
For example, a 21-year-old male with a cholesterol level of 320 may be at a significantly higher risk for heart disease. By identifying patterns in these indicators, the model will predict the likelihood of heart disease in individuals.

In conclusion, a machine learning model will be created, that evaluates key health indicators to assist healthcare facilities in assessing patient risk and developing effective preventive care strategies. Performance will be evaluated using metrics such as accuracy and precision to ensure the model's reliability.

Data Cleaning

1.Check Datatypes: Checking data types is a key step in cleaning data. It ensures that each column has the correct type, like numbers or text, which helps maintain consistency. Identifying data types also helps find errors, decide on needed changes, and perform accurate calculations.

Datatype result



The dataset analysis showed different data types in its columns:

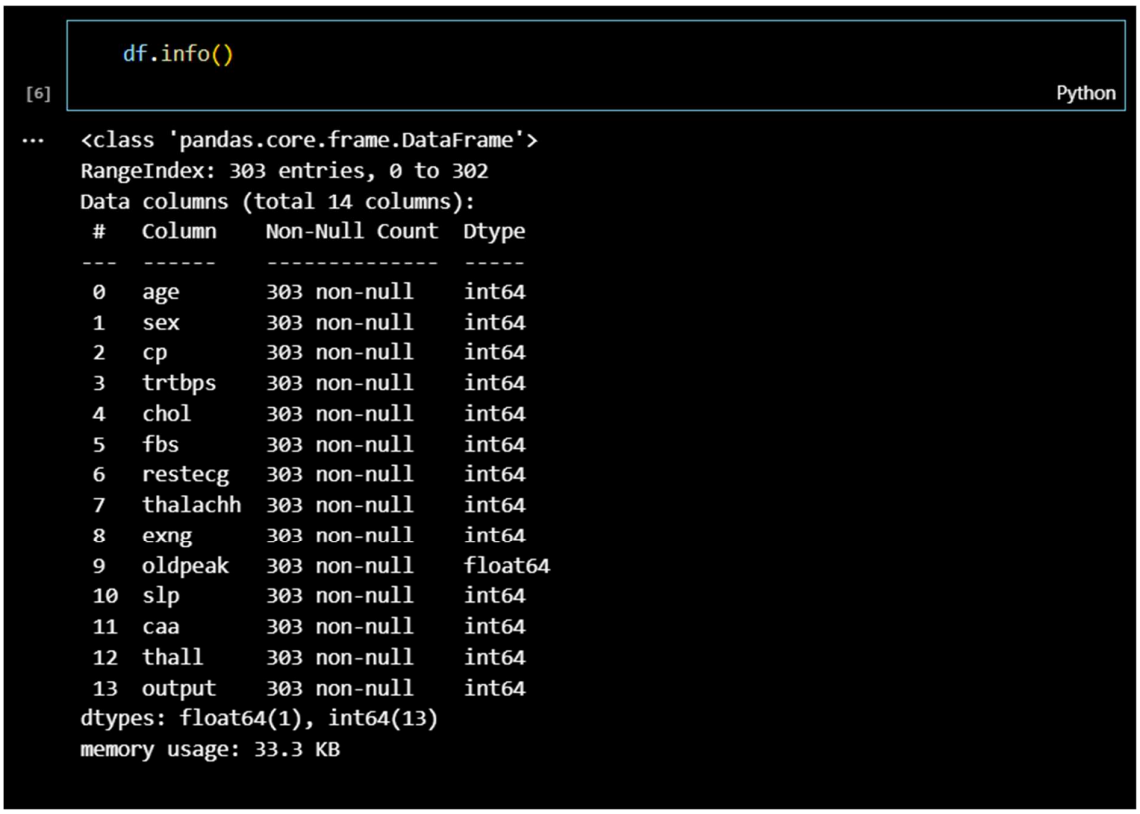
Integer Types (int64): Most columns, like age, sex, and chol, have whole numbers, which can be used for both categories and continuous data.

Float Type (float64): The oldpeak column has decimal values, representing a continuous measurement.

Object Type: The dtype column is labeled as an object type, indicating it may have non-numeric data that might need to be changed to a categorical type.

2. Checking DataFrame Information: Reviewing the DataFrame information gives a clear picture of the dataset's structure, including the number of entries, total columns, and data types for each column. This step is crucial for assessing the dataset's completeness, spotting missing values, and ensuring the data is prepared for analysis.

DataFrame Information



The DataFrame has 303 entries and 14 columns, with no missing values. The data types for the columns are:

- **Integer Types (int64):** Most columns, such as age, sex, chol, and output, contain whole numbers suitable for both categories and continuous data.
- **Float Type (float64):** The oldpeak column is the only one with decimal values, indicating a continuous measurement.

3.Change Column Names: The dataset had its columns renamed to make them clearer and easier to read. Each column now has a descriptive name that helps in understanding the data better. This change makes it easier to share results and analyse the information effectively.

Renamed Columns:

```
     age  sex  Chest Pain  resting blood pressure  cholestoral  \
0     63    1           3                     145          233
1     37    1           2                     130          250
2     41    0           1                     130          204
3     56    1           1                     120          236
4     57    0           0                     120          354
..   ...  ...         ...                     ...          ...
298   57    0           0                     140          241
299   45    1           3                     110          264
300   68    1           0                     144          193
301   57    1           0                     130          131
302   57    0           1                     130          236

     fasting blood sugar  resting electrocardiographic results  \
0                      1                                     0
1                      0                                     1
2                      0                                     0
3                      0                                     1
4                      0                                     1
..                   ...                                   ...
298                    0                                     1
299                    0                                     1
300                    1                                     1
301                    0                                     1
302                    0                                     0
...
301                      115     1      1.2    1    1      3       0
302                      174     0      0.0    1    1      2       0

[303 rows x 14 columns]
```

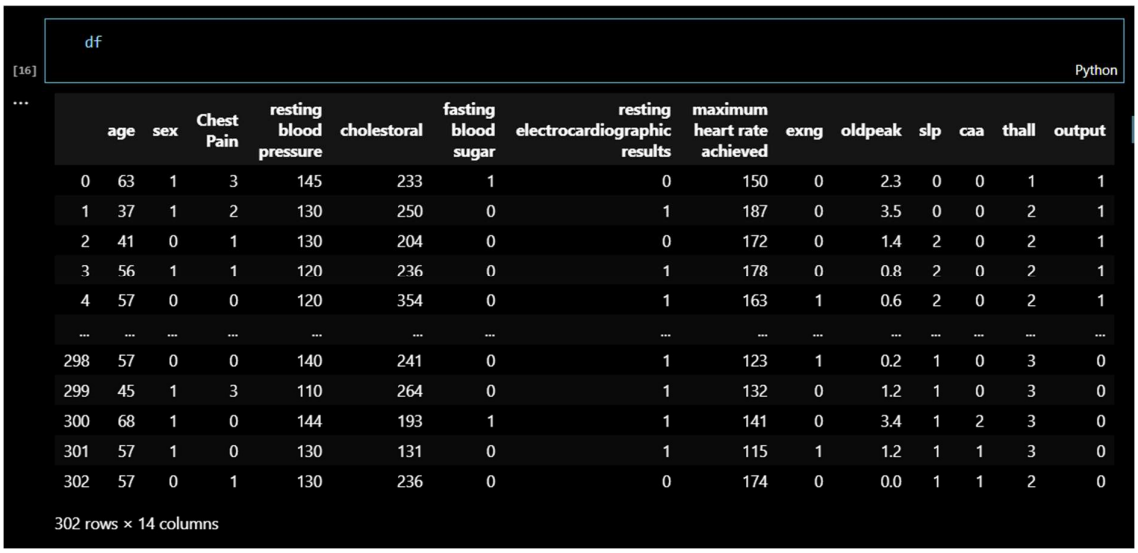
This output shows a preview of the DataFrame containing 303 rows and 14 columns. Each row corresponds to a persons health data.

Here's a breakdown of the columns data shown:

- Age: Age of the person.
- Sex: Gender of the person (0=Female, 1=male).
- Chest pain: The type of chest pain experienced.
- Resting blood pressure: Blood pressure.
- Cholesterol: Level of cholesterol.
- Fasting blood sugar: Whether its more than 120 mg/dL (0=false, 1=true**)**
- Resting electrocardiographic results: Results coded as integers resting blood pressure: Measures resting blood pressure

Displaying the dataframe

- This is the full DataFrame with the new column names being displayed, containing 303 rows of data and 14 colums, each showing health related measures for individuals.



```
[16]  df
```

Python

	age	sex	Chest Pain	resting blood pressure	cholestoral	fasting blood sugar	resting electrocardiographic results	maximum heart rate achieved	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

302 rows × 14 columns

4.Check for duplicates: The dataset was checked for duplicate rows, and it confirmed duplicates with a result of True. This suggests there may be data entry errors or repeated observations. The next steps involve identifying and removing these duplicates to ensure data integrity for accurate analysis.

```
[9]  #Check for Duplicates //done by 222119810 - Slyken Kakuva
     df.duplicated().any()                                          Python

...  True
```

- This code identifies and prints duplicate rows in the DataFrame, with reference to a duplicate found in row 164.

- The output indicates that a duplicate row is found in row 164 as shown below.

This shows that row 164 is a duplicate row in the dataset.

```
[10] #Print of the Duplicate values //done by 222119810 - Slysken Kakuva
     duplicate_rows = df[df.duplicated()]
     print(duplicate_rows)
     #Duplicate values found in row 164 //done by 222119810 - Slysken Kakuva
                                                                    Python

...       age  sex  Chest Pain  resting blood pressure  cholestoral  \
     164   38    1           2                    138          175

          fasting blood sugar  resting electrocardiographic results  \
     164                    0                                     1

          maximum heart rate achieved  exng  oldpeak  slp  caa  thall  output
     164                          173     0      0.0    2    4      2       1
```

5. Remove Duplicate Values: duplicate values were removed from the dataset, resulting in a cleaned dataset with 298 rows. This process improves data integrity by eliminating potential data entry errors and ensuring accurate analysis in the future.

```
[11] #Removal of the duplocate values and then print the result //done by 222119810 - Slysken Kakuva
     df.drop_duplicates(inplace=True)
     #Duplicate values found in row 164 have successfully been removed //done by 222119810 - Slysken Kakuva
     df                                                             Python
```

	age	sex	Chest Pain	resting blood pressure	cholestoral	fasting blood sugar	resting electrocardiographic results	maximum heart rate achieved	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

302 rows × 14 columns

6.Check Missing Values: The dataset was checked for missing values and confirmed that there are no present. This indicates a complete dataset, ensuring reliable analysis and accurate results in the future.

The output shows False for every column, meaning that there are no missing values in any of the columns.

Each column is filled with valid data.

```
[12]  df.isnull().any()                                                    Python

...   age                                  False
      sex                                  False
      Chest Pain                           False
      resting blood pressure               False
      cholestoral                          False
      fasting blood sugar                  False
      resting electrocardiographic results False
      maximum heart rate achieved          False
      exng                                 False
      oldpeak                              False
      slp                                  False
      caa                                  False
      thall                                False
      output                               False
      dtype: bool
```

- We will use the same code to check for any duplicate rows in the DataFrame.

- The result is false, showing that there are no duplicate rows remaining.

```
[13]  #Checking if all duplicates were dropped
      df.duplicated().any()                                                Python

...   False
```

7.Checking the first five rows: The first five rows of the dataset were examined, providing an overview of the data's structure and any potential anomalies. This initial review helps in understanding the dataset and planning further analysis and quality checks.

```
[14]
#to check if something is unusual //done by Kallina Joseph
df.head()
```
Python

	age	sex	Chest Pain	resting blood pressure	cholestoral	fasting blood sugar	resting electrocardiographic results	maximum heart rate achieved	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

8. Checking the Last five rows: The last five rows of the dataset were reviewed, providing a glimpse of the final entries and offering insights into the dataset's structure, as well as any potential anomalies or trends at the end.

```
[15]
#to check if something is unusual //done by Kallina Joseph
df.tail()
```
Python

	age	sex	Chest Pain	resting blood pressure	cholestoral	fasting blood sugar	resting electrocardiographic results	maximum heart rate achieved	exng	oldpeak	slp	caa	thall	output
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

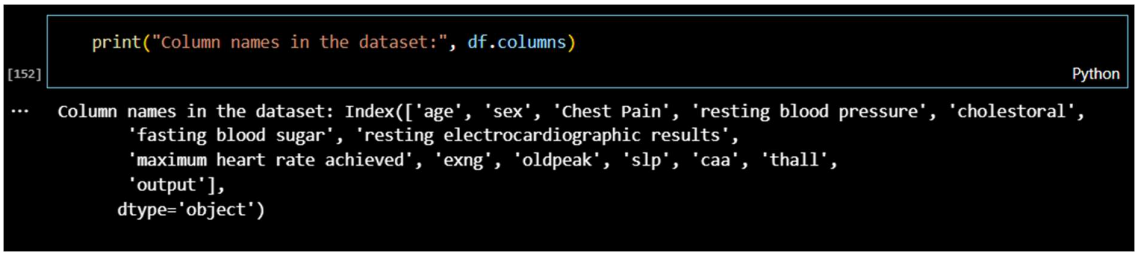
Data Structure: Like the initial rows, the last entries confirm the presence of the same attributes, including patient age, sex, chest pain type, blood pressure, cholesterol levels, and other relevant health metrics.

Consistency: The structure and format of the data remain consistent with the earlier entries, indicating uniformity throughout the dataset.

Output Values: The target variable (output) shows both 0s and 1s, suggesting a mix of patients with and without heart disease, which is essential for classification tasks.

Data Exploration

1. Print Column Names: The dataset's column names were reviewed to understand its structure and the included variables. This examination aids in recognizing the available data for analysis and informs future exploratory efforts. The columns encompass various attributes related to patient demographics and health metrics, which are essential for later analytical tasks.



2. Shape of the dataframe: The analysis shows important details about the dataset, including its size and average values for certain health measurements. There are 302 entries and 14 columns, providing a solid amount of data for examination.

The average cholesterol level is 246.5, which may indicate a higher risk of heart disease. The mean resting blood pressure is 131.6, also on the high side, suggesting it needs attention. The average maximum heart rate reached is 149.57, which helps assess heart fitness, while the average oldpeak of 1.04 could point to potential heart problems.

These results highlight key health indicators that can help identify risks among patients, showing the importance of further analysis to understand heart health better.

```
print("Shape of the data (rows,columns):",df.shape)

chol_mean = df['cholestoral'].mean()
print(f"Mean Cholesterol: {chol_mean}")

age_mean = df['age'].mean()
print(f"Mean Age: {age_mean}")

trtbps_mean = df['resting blood pressure'].mean()
print(f"Mean for resting blood pressure: {trtbps_mean}")

thalachh_mean = df['maximum heart rate achieved'].mean()
print(f"Mean for the maximum heart rate achieved: {thalachh_mean}")

oldpeak_mean = df['oldpeak'].mean()
print(f"Mean oldpeak: {oldpeak_mean}")
```

[153] Python

```
Shape of the data (rows,columns): (302, 14)
Mean Cholesterol: 246.5
Mean Age: 54.420529801324506
Mean for resting blood pressure: 131.60264900662253
Mean for the maximum heart rate achieved: 149.56953642384107
Mean oldpeak: 1.0430463576158941
```

3. Calculate and print median values: This code calculates and prints the median values for specific health indicators in a DataFrame. The median, preferred over the mean for skewed data, represents the middle value in sorted data and offers insights into typical levels for each health measure in the dataset.

The output shows the following median values: cholesterol at 240.5, age at 55.5, resting blood pressure at 130.0, maximum heart rate achieved at 152.5, and oldpeak at 0.8.

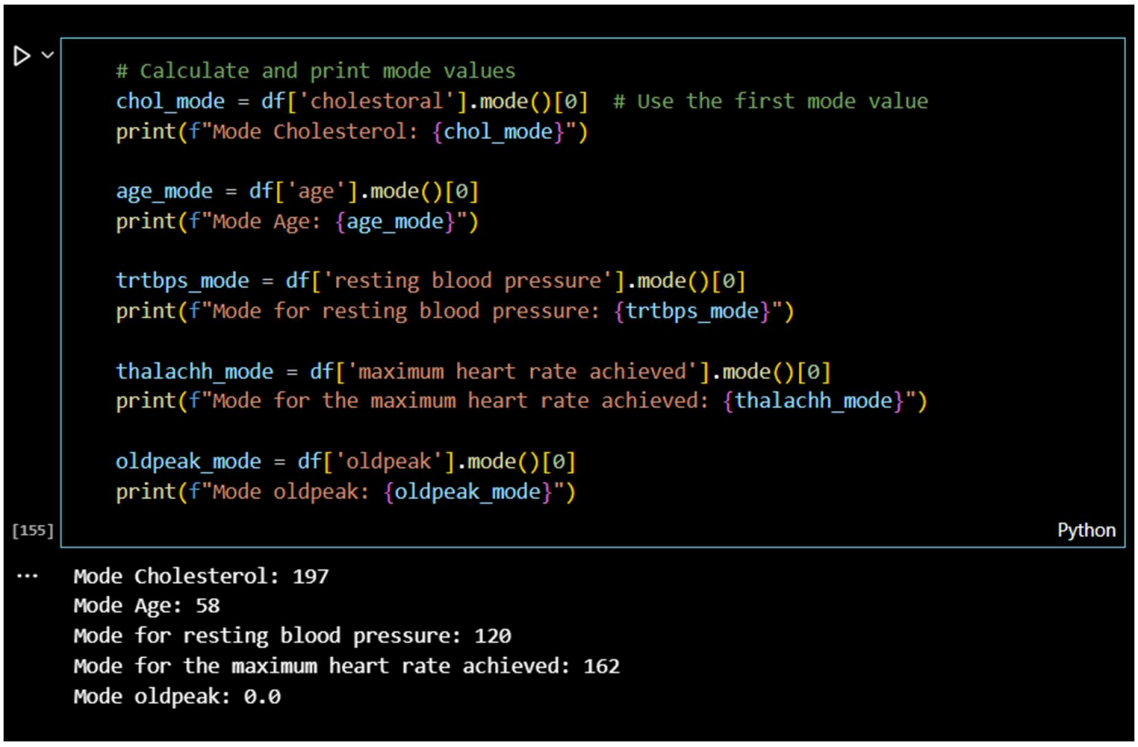
```
# Calculate and print median values
chol_median = df['cholestoral'].median()
print(f"Median Cholesterol: {chol_median}")

age_median = df['age'].median()
print(f"Median Age: {age_median}")

trtbps_median = df['resting blood pressure'].median()
print(f"Median for resting blood pressure: {trtbps_median}")

thalachh_median = df['maximum heart rate achieved'].median()
print(f"Median for the maximum heart rate achieved: {thalachh_median}")

oldpeak_median = df['oldpeak'].median()
print(f"Median oldpeak: {oldpeak_median}")
```

[154] Python

```
Median Cholesterol: 240.5
Median Age: 55.5
Median for resting blood pressure: 130.0
Median for the maximum heart rate achieved: 152.5
Median oldpeak: 0.8
```

4. Calculate and Print Mode value: This code calculates the mode values for specific columns in the DataFrame. The Mode is the value that appears the most frequently in a dataset. The mode can be used to gain insights into common characteristics in the dataset, such as the most frequent age, blood pressure level, or cholesterol reading.

The output shows the mode values for each variable: cholesterol at 197, age at 58, resting blood pressure at 120, maximum heart rate achieved at 162, and oldpeak at 0.0.



```
# Calculate and print mode values
chol_mode = df['cholestoral'].mode()[0]  # Use the first mode value
print(f"Mode Cholesterol: {chol_mode}")

age_mode = df['age'].mode()[0]
print(f"Mode Age: {age_mode}")

trtbps_mode = df['resting blood pressure'].mode()[0]
print(f"Mode for resting blood pressure: {trtbps_mode}")

thalachh_mode = df['maximum heart rate achieved'].mode()[0]
print(f"Mode for the maximum heart rate achieved: {thalachh_mode}")

oldpeak_mode = df['oldpeak'].mode()[0]
print(f"Mode oldpeak: {oldpeak_mode}")
```

[155] Python

```
Mode Cholesterol: 197
Mode Age: 58
Mode for resting blood pressure: 120
Mode for the maximum heart rate achieved: 162
Mode oldpeak: 0.0
```

5. Calculate and Print Variance values: The code calculates and prints the variance for different health-related metrics from the DataFrame df. Variance is a measure of how much the values in a dataset differ from the mean. Higher variance indicates more spread out data points.

The output displays the variance values for each specified column: cholesterol at 2678.43, age at 81.87, resting blood pressure at 308.47, maximum heart rate achieved at 524.57, and oldpeak at 1.35.

```
# Calculate and print Variance values
chol_var = df['cholestoral'].var()
print(f"Variance for Cholesterol: {chol_var}")

age_var = df['age'].var()
print(f"Variance for age: {age_var}")

trtbps_var = df['resting blood pressure'].var()
print(f"Variance for resting blood pressure: {trtbps_var}")

thalachh_var = df['maximum heart rate achieved'].var()
print(f"Variance for the maximum heart rate achieved: {thalachh_var}")

oldpeak_var = df['oldpeak'].var()
print(f"Variance for oldpeak: {oldpeak_var}")
```

[156] Python

```
Variance for Cholesterol: 2678.423588039867
Variance for age: 81.86575652900926
Variance for resting blood pressure: 308.4728168797166
Variance for the maximum heart rate achieved: 524.5715605817254
Variance for oldpeak: 1.3489714197707423
```

6. Calculate and print standard deviation values: The code calculates and prints the standard deviation for different health-related metrics from the DataFrame. Standard deviation is a measure of the amount of variation or dispersion in a set of values. A lower standard deviation indicates that the values tend to be close to the mean, while a higher standard deviation indicates that the values are spread out over a wider range.

The output presents the standard deviation values for each specified column: cholesterol at 51.75, age at 9.05, resting blood pressure at 17.56, maximum heart rate achieved at 22.90, and oldpeak at 1.16.

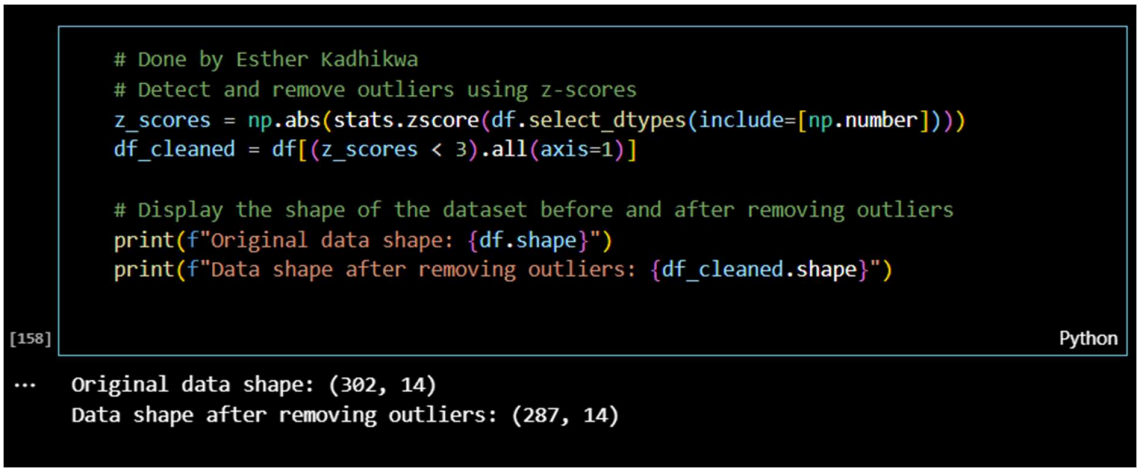
```
# Calculate and print Standard deviation values
chol_std = df['cholestoral'].std()
print(f"Standard deviation for Cholesterol: {chol_std}")

age_std = df['age'].std()
print(f"Standard deviation for age: {age_std}")

trtbps_std = df['resting blood pressure'].std()
print(f"Standard deviation for resting blood pressure: {trtbps_std}")

thalachh_std = df['maximum heart rate achieved'].std()
print(f"Standard deviation for the maximum heart rate achieved: {thalachh_std}")

oldpeak_std = df['oldpeak'].std()
print(f"Standard deviation for oldpeak: {oldpeak_std}")
```

[157] Python

```
Standard deviation for Cholesterol: 51.75348865574056
Standard deviation for age: 9.047969746247457
Standard deviation for resting blood pressure: 17.56339423003756
Standard deviation for the maximum heart rate achieved: 22.903527251969845
Standard deviation for oldpeak: 1.1614522890634562
```

7. Detect and remove outliers using z-score: The code effectively removes outliers from the dataset using Z-scores, ensuring that the remaining data points are within a reasonable range of the mean. This is useful for improving the quality of data analysis by eliminating extreme values that could skew the results.

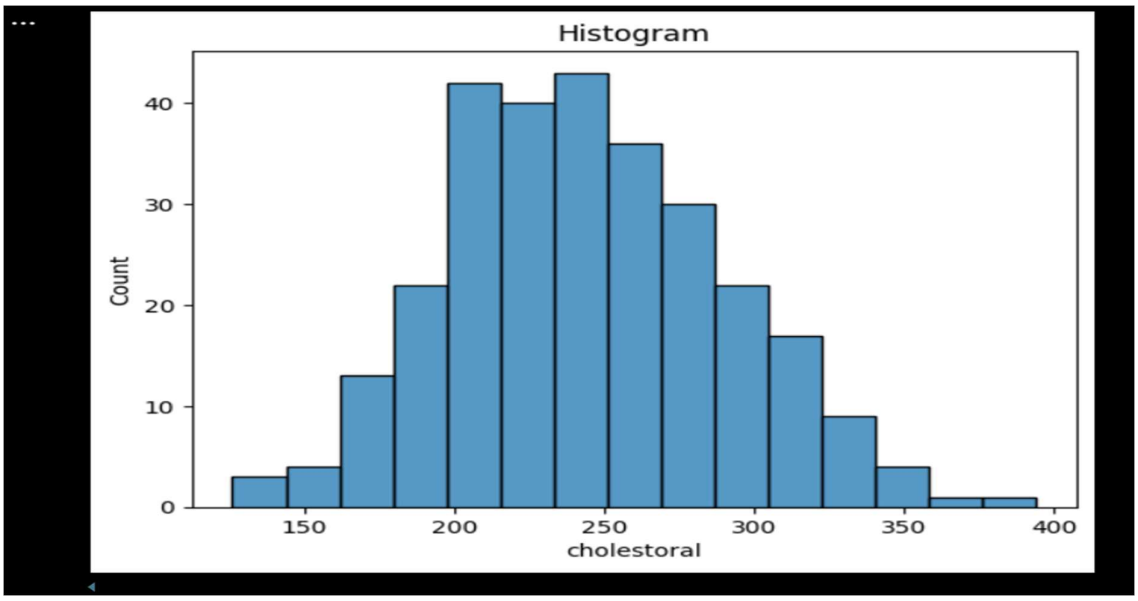
The output indicates that the original dataset had a shape of (302, 14), while the shape after removing outliers is (287, 14).



8. Histograms: The code visualizes the distribution of various health-related metrics using histograms and calculates statistical measures (skewness and kurtosis) to understand the data's distribution characteristics. Skewness measures the asymmetry of the data distribution, while kurtosis measures the "tailedness" of the distribution.

```python
sns.histplot(df_cleaned['cholestoral'], kde=False)
plt.title('Histogram')
plt.show()

sns.histplot(df_cleaned['age'], kde=False)
plt.title('Histogram')
plt.show()

sns.histplot(df_cleaned['resting blood pressure'], kde=False)
plt.title('Histogram')
plt.show()

sns.histplot(df_cleaned['maximum heart rate achieved'], kde=False)
plt.title('Histogram')
plt.show()

sns.histplot(df_cleaned['oldpeak'], kde=False)
plt.title('Histogram')
plt.show()

columns_to_analyze = ['cholestoral', 'age', 'resting blood pressure', 'maximum heart rate achieved', 'oldpeak']

print(f"Skewness: {skew(df_cleaned[columns_to_analyze])}")
print(f"Kurtosis: {kurtosis(df_cleaned[columns_to_analyze])}")
```

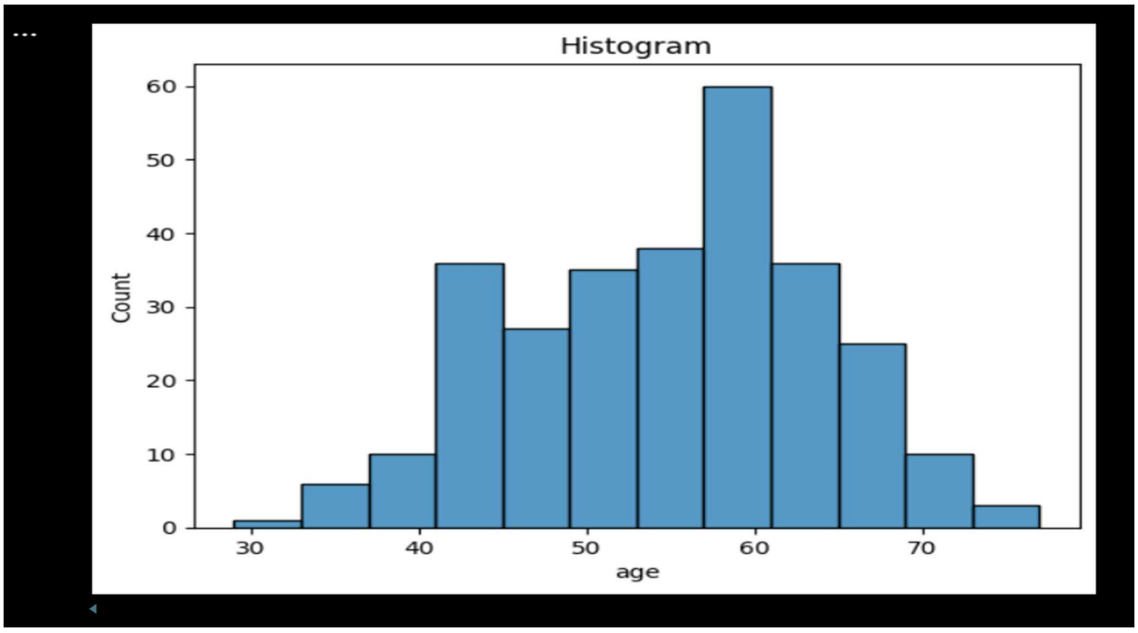
[159] Python



X-axis: Cholesterol, showing the range of cholesterol level

Y-axis: Count, showing the number of individuals within cholesterol range.

In this histogram, the most common cholesterol levels are between 200 and 250.



X-axis: Age, Showing the range of ages from 20 to 80.

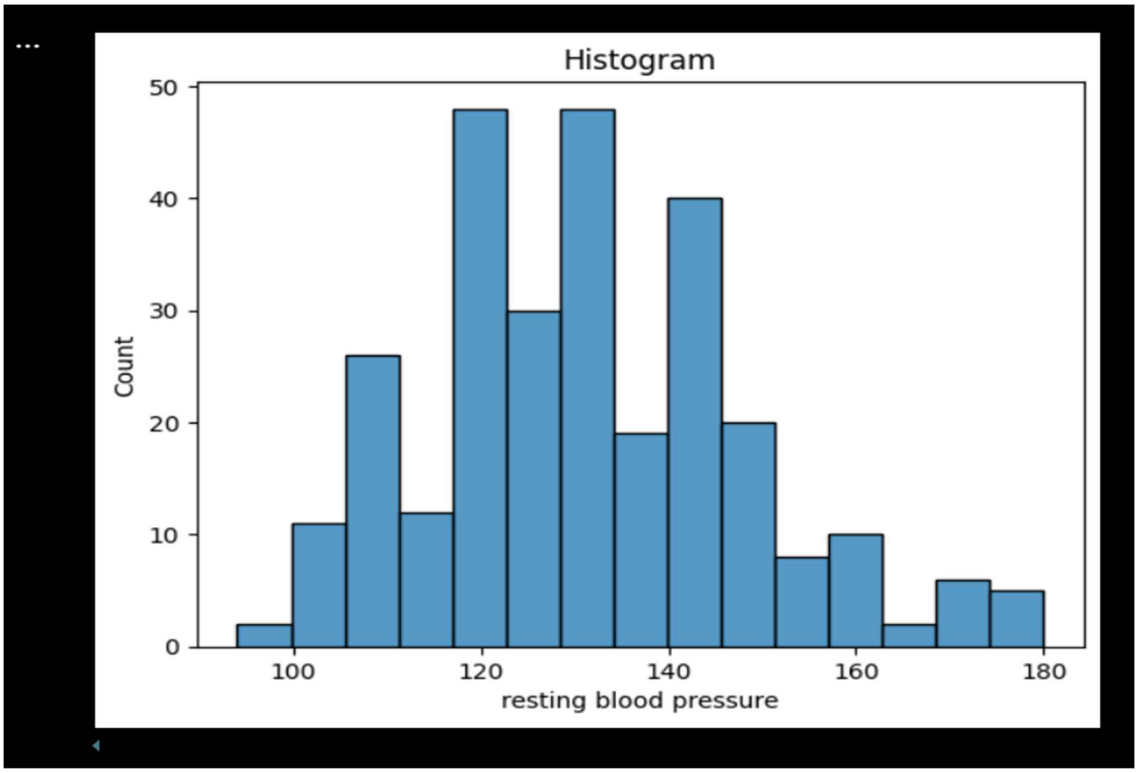
Y-axis: Count, showing the number of individuals within each age range.

From this histogram, the most common age range is around 60.

X-axis: Resting blood pressure, it ranges from 100 to 180.

Y-axis: count, showing the number of occurrences within each range.

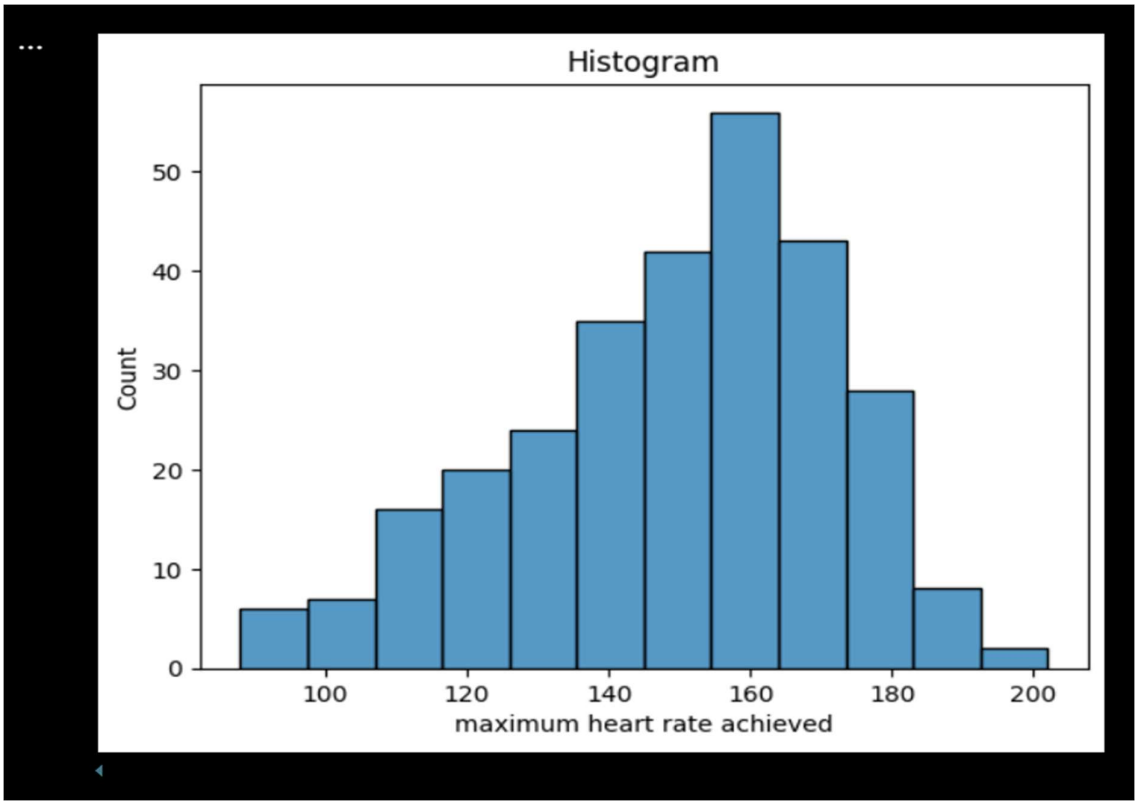
This histogram has two prominent peaks, and the most resting blood pressure values lie within those specific ranges



X-axis: Maximum heart rate achieved, ranging from 80 to 200.

Y-axis: count, indicating the number of occurrences with each range, ranging from 0 to 50.

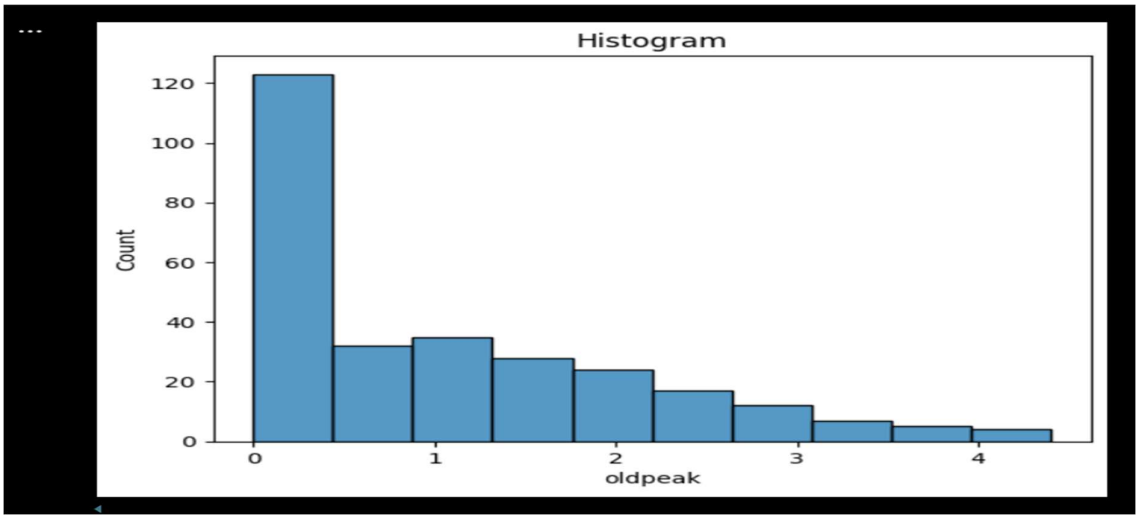
The histogram shows that the most common maximum heart rate values are around 140 to 160.



X-axis: Oldpeak, ranging from 0 to 4

Y-axis: count, indicating the number of occurrences within each range.

This histogram shows that the most common oldpeak value is around 0.



The skewness values indicate that the first value (0.24345062) shows a slight positive skew, the second value (-0.1383566) reflects a slight negative skew, the third value (0.527394) indicates a moderate positive skew, the fourth value (-0.47830412) represents a moderate negative skew, and the fifth value (0.97910045) demonstrates a strong positive skew.

The kurtosis values show that the first value (-0.14139822) indicates slightly lower than normal kurtosis (platykurtic), the second value (-0.55057245) reflects lower than normal kurtosis (platykurtic), the third value (0.21685942) signifies slightly higher than normal kurtosis (leptokurtic), the fourth value (-0.3430916) indicates lower than normal kurtosis (platykurtic), and the fifth value (1.4636728) demonstrates significantly higher than normal kurtosis (leptokurtic).

```
Skewness: [ 0.24430542 -0.18335668  0.5279344  -0.47138412  0.97910445]
Kurtosis: [-0.14492382 -0.55042754  0.21859423 -0.3429186   0.1436728 ]
```

9. Boxplot: The code renames specific columns in the DataFrame for clarity and then creates box plots for these columns to visualize their distributions. Box plots are useful for showing the spread and skewness of the data, as well as identifying outliers.

```
# Specify the columns to plot
columns_to_plot = ['age', 'resting blood pressure', 'cholestoral', 'maximum heart rate achieved', 'oldpeak']

renamed_columns = ['age', 'rbp', 'Cholestoral', 'mhra', 'oldpeak']

# Rename the columns
df_renamed = df_cleaned[columns_to_plot].rename(columns=dict(zip(columns_to_plot, renamed_columns)))

# Create box plots for the selected columns
df_renamed.boxplot()
plt.title('Box Plot for Selected Columns')
plt.xlabel('Columns')
plt.ylabel('Values')
plt.show()
```

160]

Python

X-axis labelled with different columns, age, rbp (resting blood pressure), cholesterol, mhra (maximum heart rate achieved), and oldpeak

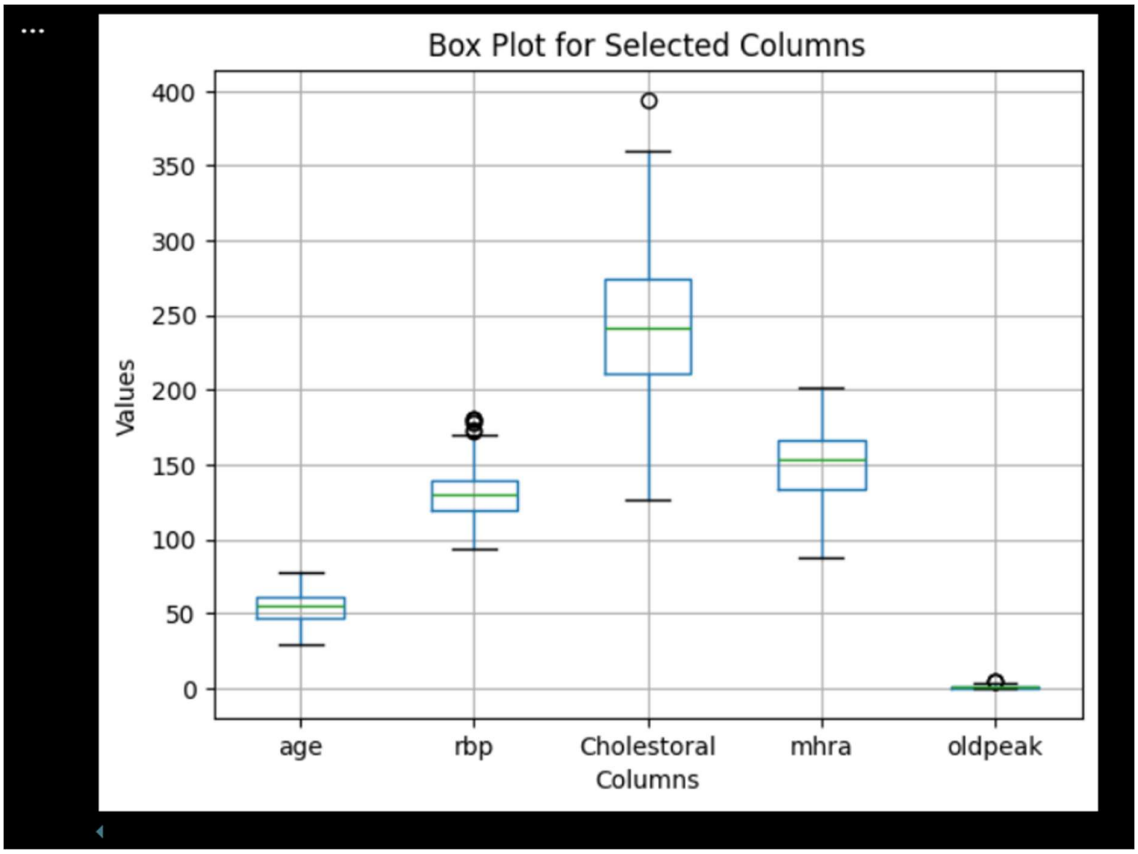
Y-axis labelled, values ranging from 0 to 400.

Median: The line inside the box, representing the middle value of the data.

Interquartile Range: The length of a box, showing the range between the first quartile and the third quartile.

Whiskers: The line extending from the box, indicating the upper and lower quartiles.

Outliers: Dots outside the whiskers, representing data points that fall significantly outside the range.



10. One-sample t-test: The code performs a one-sample t-test on several columns of the DataFrame to determine whether the means of those columns significantly differ from a specified value.

The t-statistic value represents the ratio of the difference between the sample mean and the mean to the standard error.

P-value indicates the probability of observing a t-statistc as extreme as the one calculated, assuming that the null hypothesis is true.

```
[165]
data = df_cleaned[columns_to_analyze].dropna()
t_stat, p_value = stats.ttest_1samp(data, 50)
print(f"T-statistic: {t_stat}, P-value: {p_value}")
```
Python

```
T-statistic: [-4899.81114387     8.07544129 -4250.235267       74.97823004
  -768.05975927], P-value: [0.00000000e+000 1.88977060e-014 0.00000000e+000 4.27774725e-190
 0.00000000e+000]
```

11. The output of this code displays the first five rows of the DataFrame.

Which contains cleaned data.

```
[161]
df_cleaned.head()
```
Python

	age	sex	Chest Pain	resting blood pressure	cholestoral	fasting blood sugar	resting electrocardiographic results	maximum heart rate achieved	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

12. Convert specific columns to categorical data types: The code converts specific columns in the DataFrame to the ‘category’ data type, which can save memory and improve performance for categorical data. The output confirms that the conversions were successful by displaying the updated data types of each column.

The output shows that the data types of the columns remain as integers instead of changing to category

```
[162]
# Convert specific columns to categorical data types
df_cleaned.loc[:, 'sex'] = df_cleaned['sex'].astype('category')
df_cleaned.loc[:, 'Chest Pain'] = df_cleaned['Chest Pain'].astype('category')
df_cleaned.loc[:, 'thall'] = df_cleaned['thall'].astype('category')
df_cleaned.loc[:, 'exng'] = df_cleaned['exng'].astype('category')
df_cleaned.loc[:, 'caa'] = df_cleaned['caa'].astype('category')
df_cleaned.loc[:, 'fasting blood sugar'] = df_cleaned['fasting blood sugar'].astype('category')
df_cleaned.loc[:, 'resting electrocardiographic results'] = df_cleaned['resting electrocardiographic results'].astype('category')

# Display the data types after conversion
print(df_cleaned.dtypes)
```
Python

```
age                                 int64
sex                                 int64
Chest Pain                          int64
resting blood pressure              int64
cholestoral                         int64
fasting blood sugar                 int64
resting electrocardiographic results    int64
maximum heart rate achieved         int64
exng                                int64
oldpeak                           float64
slp                                 int64
caa                                 int64
thall                               int64
output                              int64
dtype: object
```

13. Normalize the resting blood pressure and cholesterol columns: The code normalizes the ‘resting blood pressure’ and ‘cholesterol’ columns in the DataFrame df_cleaned using the MinMaxScaler, which scales the values to a range between 0 and 1. This is a common preprocessing step in machine learning to ensure that all features contribute equally to the model.

```
# Normalize the 'trtbps' (resting blood pressure) and 'chol' (cholesterol) columns
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df_cleaned.loc[:,['resting blood pressure', 'cholestoral']] = scaler.fit_transform(df_cleaned[['resting blood pressure', 'cholestoral']])

# Display the first few rows of the normalized data
print(df_cleaned[['resting blood pressure', 'cholestoral']].head())
```

[163] Python

The output shows the normalized values for ‘resting blood pressure’ and ‘cholesterol’ columns, along with a warning about deprecated behavior in pandas and an error related to indexing. These messages indicate potential issues in the code that may need to be addressed to ensure compatibility with future versions of pandas and to resolve the indexing problem.

The MinMaxScaler returns values between 0 and 1, which are floating numbers, but the columns you sre assigning them to were originally intergers.

```
...    resting blood pressure  cholestoral
0                   0.593023     0.399254
1                   0.418605     0.462687
2                   0.418605     0.291045
3                   0.302326     0.410448
4                   0.302326     0.850746
C:\Users\lkamu\AppData\Local\Temp\ipykernel_15740\4006629059.py:5: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise in a future error of pandas. Value
 0.53488372 0.30232558 0.90697674 0.65116279 0.53488372 0.41860465
 0.41860465 0.18604651 0.65116279 0.30232558 0.30232558 0.65116279
 0.65116279 0.53488372 0.47674419 0.41860465 0.53488372 0.65116279
 0.53488372 0.76744186 0.65116279 0.18604651 0.41860465 0.12790698
 0.30232558 0.41860465 0.36046512 0.36046512 0.55813953 0.47674419
 0.65116279 0.70930233 0.76744186 0.53488372 0.41860465 0.11627907
 0.41860465 0.53488372 0.30232558 0.53488372 0.51162791 0.51162791
 0.41860465 0.30232558 0.41860465 0.1627907  0.47674419 0.46511628
 0.3255814  0.24418605 0.27906977 0.39534884 0.18604651 0.1627907
 0.27906977 0.47674419 0.53488372 0.51162791 0.06976744 0.41860465
 0.30232558 0.34883721 0.30232558 0.         0.41860465 0.53488372
 0.3255814  0.47674419 0.36046512 0.53488372 0.39534884 0.12790698
 0.20930233 0.39534884 0.09302326 0.6744186  0.09302326 0.27906977
 0.08139535 0.18604651 0.06976744 0.34883721 0.44186047 0.44186047
 0.20930233 0.55813953 0.53488372 0.1627907  0.41860465 0.41860465
 0.62790698 0.97674419 0.53488372 0.30232558 0.40697674 0.30232558
 0.76744186 0.51162791 0.30232558 0.18604651 1.         0.65116279
 0.53488372 0.18604651 0.41860465 0.30232558 0.41860465 0.30232558
 0.12790698 0.51162791 0.41860465 0.51162791 0.20930233 0.1627907
 0.         0.27906977 0.20930233 0.6744186  0.48837209 0.30232558
 0.76744186 0.46511628 0.30232558 0.18604651 0.37209302 0.41860465
 0.30232558 0.39534884 0.18604651 0.39534884 0.30232558 0.24418605
 0.30232558 0.13953488 0.53488372 0.72093023 0.27906977 0.65116279
 0.30232558 0.41860465 0.76744186 0.20930233 0.88372093 0.60465116
...
 0.29104478 0.39552239 0.77985075 0.29477612 0.28731343 0.71641791
 0.36940299 0.32089552 0.16044776 0.22761194 0.26492537 0.18656716
 0.42910448 0.51492537 0.25       0.01865672 0.41044776]' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
  df_cleaned.loc[:,['resting blood pressure', 'cholestoral']] = scaler.fit_transform(df_cleaned[['resting blood pressure', 'cholestoral']])
```

14. One-Hot encoding Categorical Columns: The code converts the 'Chest Pain' and 'thall' columns in the DataFrame df_cleaned into binary columns using one-hot encoding. This is a common preprocessing step in machine learning to convert categorical data into a numerical format that algorithms can work with. The output displays the first few rows of the modified DataFrame to verify the transformation.

```
# One-hot encode categorical columns like 'cp' and 'thall'
df_encoded = pd.get_dummies(df_cleaned, columns=['Chest Pain', 'thall'], drop_first=True)

# Display the first few rows of the dataset after encoding
print(df_encoded.head())
```

[164] Python

This is the result of of one-hot encoding the columns chest pain and thall in the dataset.

The binary columns chest pain_1, chest pain_2, chest pain_3, thall_2, and thall_3 replace the categorical values with numbers that machine learning can work with.

Each row now has a representation of chest pain and thall results in a binary format, where true or false indicates the presence of that category.

```
...    age  sex  resting blood pressure  cholestoral  fasting blood sugar  \
0   0   63    1                0.593023     0.399254                    1
1   1   37    1                0.418605     0.462687                    0
2   2   41    0                0.418605     0.291045                    0
3   3   56    1                0.302326     0.410448                    0
4   4   57    0                0.302326     0.850746                    0

    resting electrocardiographic results  maximum heart rate achieved  exng  \
0                                     0                          150     0
1                                     1                          187     0
2                                     0                          172     0
3                                     1                          178     0
4                                     1                          163     1

    oldpeak  slp  caa  output  Chest Pain_1  Chest Pain_2  Chest Pain_3  \
0       2.3    0    0       1         False         False          True
1       3.5    0    0       1         False          True         False
2       1.4    2    0       1          True         False         False
3       0.8    2    0       1          True         False         False
4       0.6    2    0       1         False         False         False

   thall_2  thall_3
0    False    False
1     True    False
2     True    False
3     True    False
4     True    False
```

15. Scatter Plot: The code creates a scatter plot to visualize the relationship between age and cholesterol levels. It sets up the plot with appropriate labels and titles, and then displays it. This type of plot is useful for identifying trends or correlations between two variables.

```
# scatter plot showing Cholestterol correlation with age//Done By 222119810 Kakuva Slysken
plt.figure(figsize=(8, 6))
plt.scatter(df['age'], df['cholestoral'], alpha=1.0)  # Adjusted alpha for clearer visibility
plt.title('Cholesterol vs. Age Correlation')
plt.xlabel('Age')
plt.ylabel('Cholesterol')
plt.show()
```

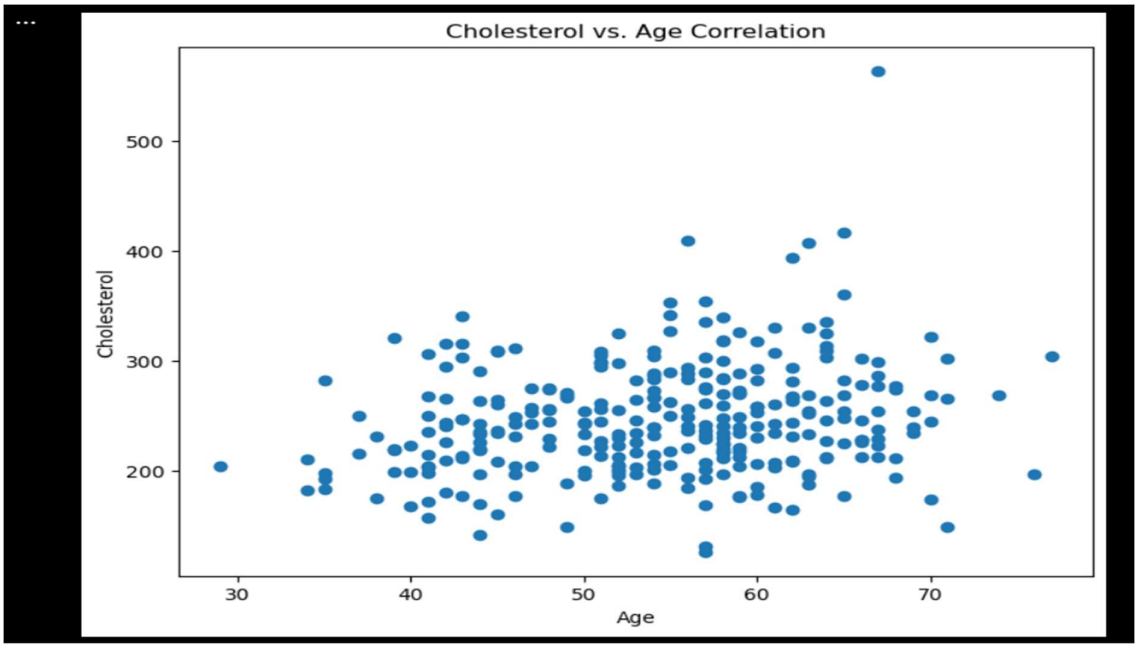
[72] Python

X-axis labelled Age; it ranges from 20 to 80.

Y-axis labelled cholesterol; it ranges from 0 to 500.

The scatter plot is used to visualize the relationship between age and cholesterol levels. This plot shows high concentrations of data where the age range if 40 to 60 and cholesterol levels between 200 to 300.

But there appears to be no strong correlation between age and cholesterol levels.



16. The Scatter Plot matrix: The code creates a scatter plot matrix to visualize the pairwise relationships between the specified features (‘age’, ‘sex’, ‘Chest Pain’, ‘cholestoral’, ‘maximum heart rate achieved’). This type of plot is useful for identifying correlations and patterns between multiple variables in a dataset.

```
[73]
# Scatter Plot Matrix //Done By 222119810 Kakuva Slysken
features = ['age', 'sex', 'Chest Pain','cholestoral','maximum heart rate achieved']  # List of features in graph
scatter_matrix(df[features], figsize=(12, 10), alpha=0.7, diagonal='hist',
hist_kwds={'bins': 10}, range_padding=0.05)  # histogram keywords to plot
plt.suptitle('Scatter Plot Matrix for Heart Disease Risk Factors Correlations', fontsize=14)  # title
plt.tight_layout()  # Space between plot graphs
plt.show()
                                                                                                    Python
```

Scatter plot matrix for Heart Disease Risk Factors correlation

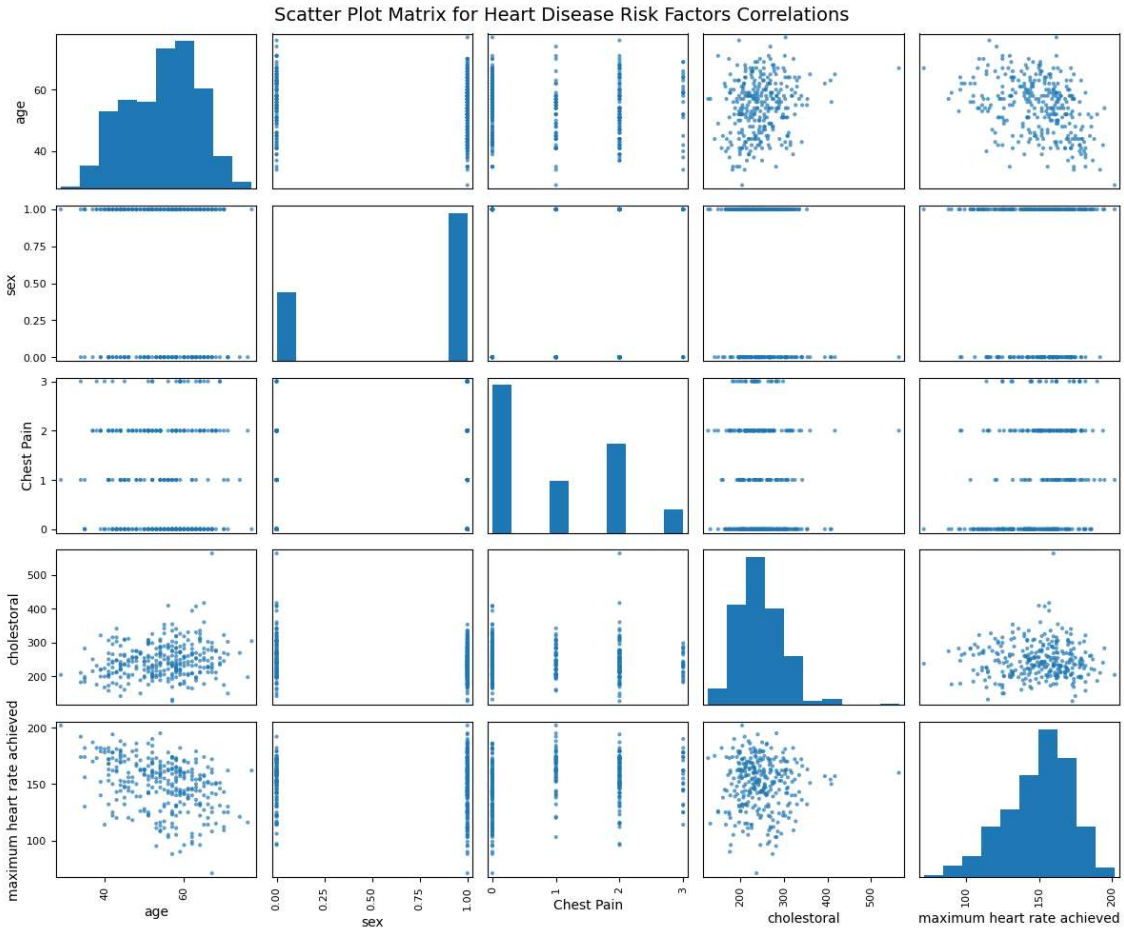
Variables: The matrix includes several variables related to heart disease risk, such as age, resting blood pressure, cholesterol, fasting blood sugar, maximum heart rate achieved, and chest pain.

Diagonal Plots: The diagonal line across the matrix shows the histograms for each variable, indicating the distribution of values for that variable.

Off-Diagonal Plots: Each off-diagonal plot shows the relationship between two variables.

Histograms provide a quick view of the distribution of each variable.

Scatter Plots help identify patterns of correlations between different pairs of variables.


Scatter Plot Matrix for Heart Disease Risk Factors Correlations
age
sex
Chest Pain
cholestoral
maximum heart rate achieved