Import the required libraries

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

Read the dataset

```
df=pd.read_csv("D://Intership//Feb months project//Heart Disease
data.csv")
```

This will display the firstly few rows

```
df.head()
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
slope  \
0   52    1   0       125   212    0        1      168      0      1.0
2
1   53    1   0       140   203    1        0      155      1      3.1
0
2   70    1   0       145   174    0        1      125      1      2.6
0
3   61    1   0       148   203    0        1      161      0      0.0
2
4   62    0   0       138   294    1        1      106      0      1.9
1

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
```

TO check the null value present in dataset

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

```
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
```

```
df.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
```

```
df.describe()
```

```
                age          sex           cp     trestbps         chol
\
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.00000

mean     54.434146     0.695610     0.942439   131.611707   246.00000

std       9.072290     0.460373     1.029641    17.516718    51.59251

min      29.000000     0.000000     0.000000    94.000000   126.00000

25%      48.000000     0.000000     0.000000   120.000000   211.00000

50%      56.000000     1.000000     1.000000   130.000000   240.00000

75%      61.000000     1.000000     2.000000   140.000000   275.00000

max      77.000000     1.000000     3.000000   200.000000   564.00000
```

```
                fbs      restecg      thalach        exang      oldpeak
\
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000

mean      0.149268     0.529756   149.114146     0.336585     1.071512

std       0.356527     0.527878    23.005724     0.472772     1.175053

min       0.000000     0.000000    71.000000     0.000000     0.000000

25%       0.000000     0.000000   132.000000     0.000000     0.000000

50%       0.000000     1.000000   152.000000     0.000000     0.800000

75%       0.000000     1.000000   166.000000     1.000000     1.800000

max       1.000000     2.000000   202.000000     1.000000     6.200000


              slope           ca         thal       target
count  1025.000000  1025.000000  1025.000000  1025.000000
mean      1.385366     0.754146     2.323902     0.513171
std       0.617755     1.030798     0.620660     0.500070
min       0.000000     0.000000     0.000000     0.000000
25%       1.000000     0.000000     2.000000     0.000000
50%       1.000000     0.000000     2.000000     1.000000
75%       2.000000     1.000000     3.000000     1.000000
max       2.000000     4.000000     3.000000     1.000000

df.corr()

                age       sex        cp  trestbps      chol
fbs  \
age        1.000000 -0.103240 -0.071966  0.271121  0.219823  0.121243

sex       -0.103240  1.000000 -0.041119 -0.078974 -0.198258  0.027200

cp        -0.071966 -0.041119  1.000000  0.038177 -0.081641  0.079294

trestbps   0.271121 -0.078974  0.038177  1.000000  0.127977  0.181767

chol       0.219823 -0.198258 -0.081641  0.127977  1.000000  0.026917

fbs        0.121243  0.027200  0.079294  0.181767  0.026917  1.000000

restecg   -0.132696 -0.055117  0.043581 -0.123794 -0.147410 -0.104051

thalach   -0.390227 -0.049365  0.306839 -0.039264 -0.021772 -0.008866
```

```
exang     0.088163  0.139157 -0.401513  0.061197  0.067382  0.049261
oldpeak   0.208137  0.084687 -0.174733  0.187434  0.064880  0.010859
slope    -0.169105 -0.026666  0.131633 -0.120445 -0.014248 -0.061902
ca        0.271551  0.111729 -0.176206  0.104554  0.074259  0.137156
thal      0.072297  0.198424 -0.163341  0.059276  0.100244 -0.042177
target   -0.229324 -0.279501  0.434854 -0.138772 -0.099966 -0.041164


            restecg   thalach     exang   oldpeak     slope
ca  \
age      -0.132696 -0.390227  0.088163  0.208137 -0.169105  0.271551
sex      -0.055117 -0.049365  0.139157  0.084687 -0.026666  0.111729
cp        0.043581  0.306839 -0.401513 -0.174733  0.131633 -0.176206
trestbps -0.123794 -0.039264  0.061197  0.187434 -0.120445  0.104554
chol     -0.147410 -0.021772  0.067382  0.064880 -0.014248  0.074259
fbs      -0.104051 -0.008866  0.049261  0.010859 -0.061902  0.137156
restecg   1.000000  0.048411 -0.065606 -0.050114  0.086086 -0.078072
thalach   0.048411  1.000000 -0.380281 -0.349796  0.395308 -0.207888
exang    -0.065606 -0.380281  1.000000  0.310844 -0.267335  0.107849
oldpeak  -0.050114 -0.349796  0.310844  1.000000 -0.575189  0.221816
slope     0.086086  0.395308 -0.267335 -0.575189  1.000000 -0.073440
ca       -0.078072 -0.207888  0.107849  0.221816 -0.073440  1.000000
thal     -0.020504 -0.098068  0.197201  0.202672 -0.094090  0.149014
target    0.134468  0.422895 -0.438029 -0.438441  0.345512 -0.382085


               thal    target
age       0.072297 -0.229324
sex       0.198424 -0.279501
cp       -0.163341  0.434854
trestbps  0.059276 -0.138772
chol      0.100244 -0.099966
fbs      -0.042177 -0.041164
```

```
restecg  -0.020504  0.134468
thalach  -0.098068  0.422895
exang     0.197201 -0.438029
oldpeak   0.202672 -0.438441
slope    -0.094090  0.345512
ca        0.149014 -0.382085
thal      1.000000 -0.337838
target   -0.337838  1.000000
```

```
df.tail()
```

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang
oldpeak  \
1020   59    1   1       140   221    0        1      164      1
0.0
1021   60    1   0       125   258    0        0      141      1
2.8
1022   47    1   0       110   275    0        0      118      1
1.0
1023   50    0   0       110   254    0        0      159      0
0.0
1024   54    1   0       120   188    0        1      113      0
1.4

      slope  ca  thal  target
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0
```

```
df.shape
```

```
(1025, 14)
```

```
df.describe
```

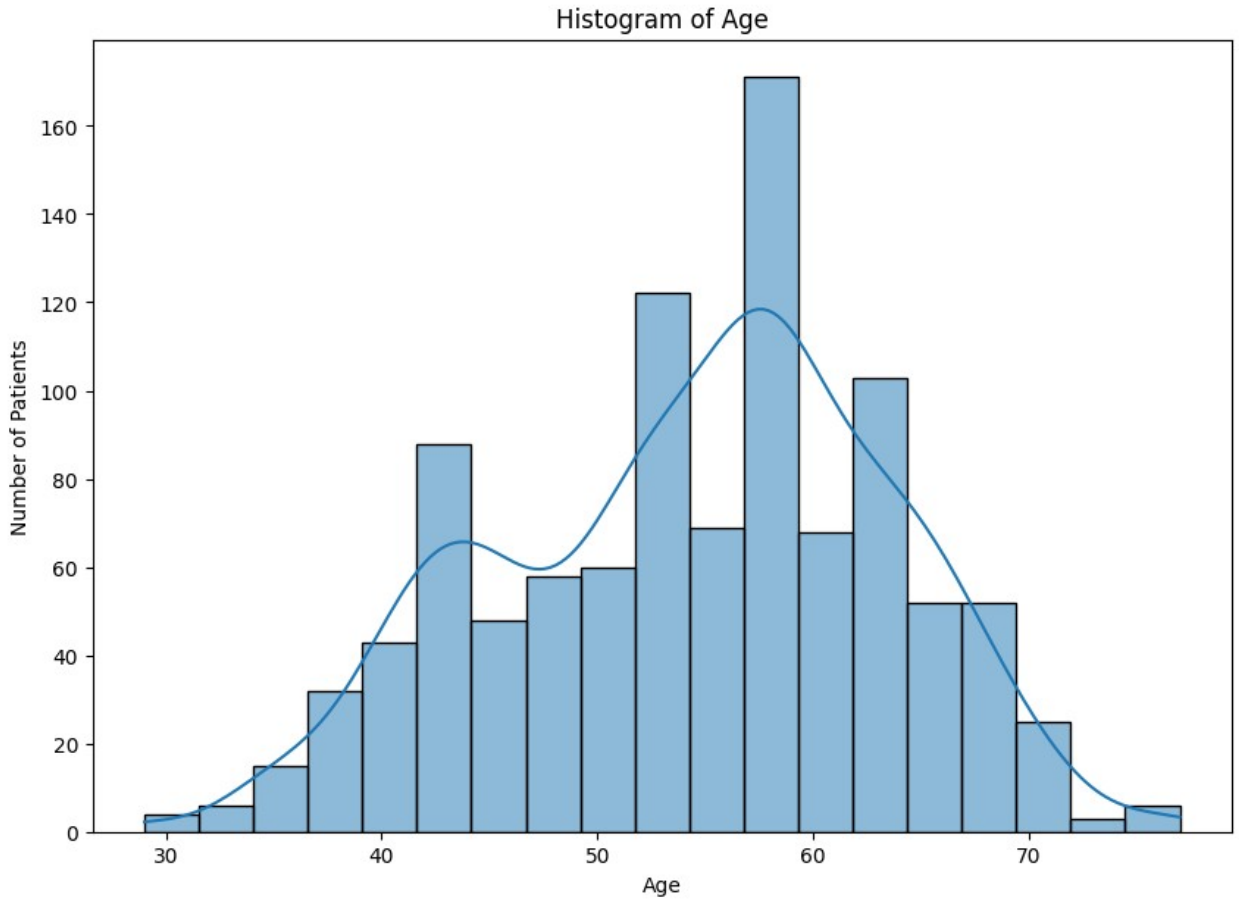
```
<bound method NDFrame.describe of       age  sex  cp  trestbps  chol
fbs  restecg  thalach  exang  oldpeak  \
0      52    1   0       125   212    0        1      168      0
1.0
1      53    1   0       140   203    1        0      155      1
3.1
2      70    1   0       145   174    0        1      125      1
2.6
3      61    1   0       148   203    0        1      161      0
0.0
4      62    0   0       138   294    1        1      106      0
1.9
...   ...  ...  ..       ...   ...  ...      ...      ...    ...
...
```

```
1020   59    1   1       140   221    0        1       164       1
0.0
1021   60    1   0       125   258    0        0       141       1
2.8
1022   47    1   0       110   275    0        0       118       1
1.0
1023   50    0   0       110   254    0        0       159       0
0.0
1024   54    1   0       120   188    0        1       113       0
1.4

      slope  ca  thal  target
0         2   2     3       0
1         0   0     3       0
2         0   0     3       0
3         2   1     3       0
4         1   3     2       0
...     ...  ..   ...     ...
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0

[1025 rows x 14 columns]>
```

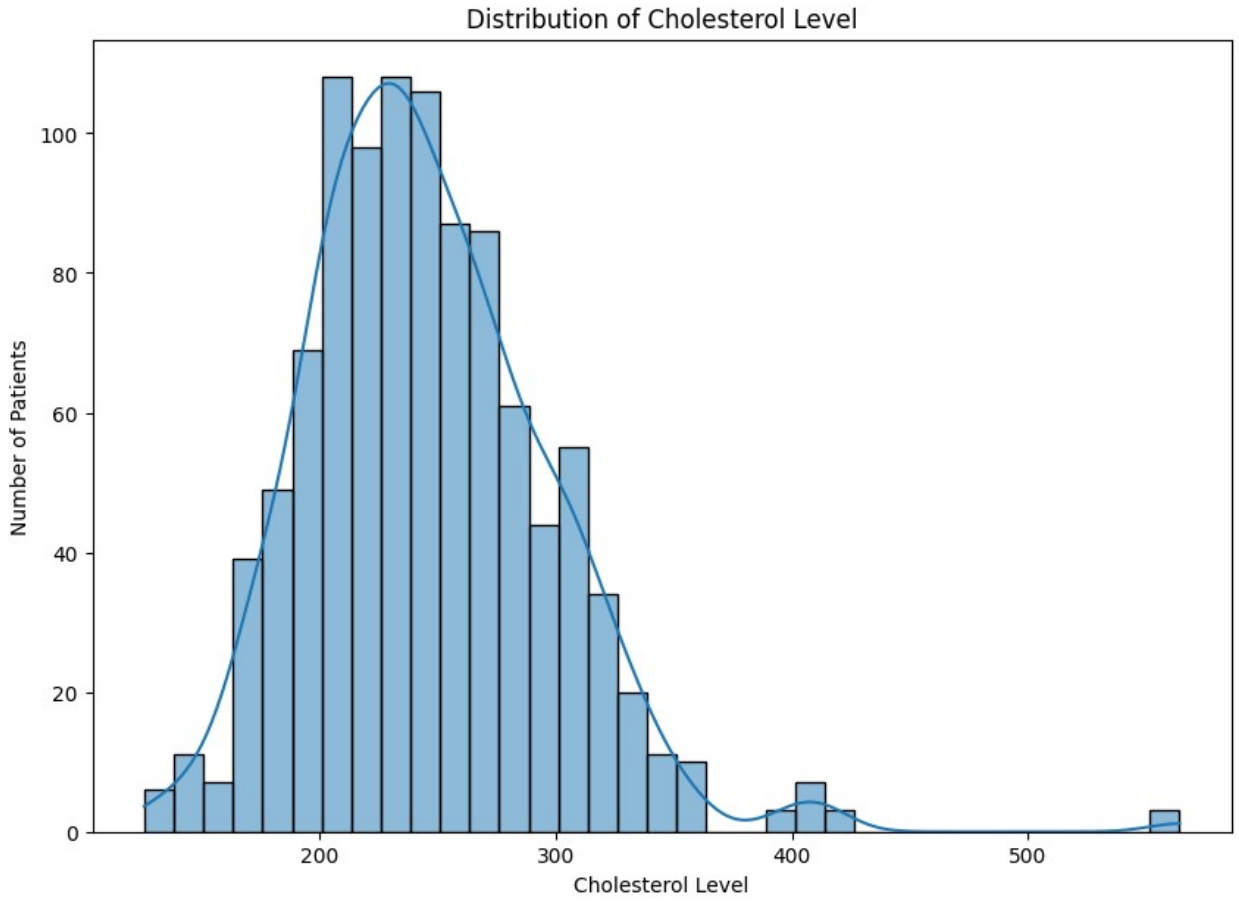
Exploratory Data Analysis (EDA)

Univariate Analysis

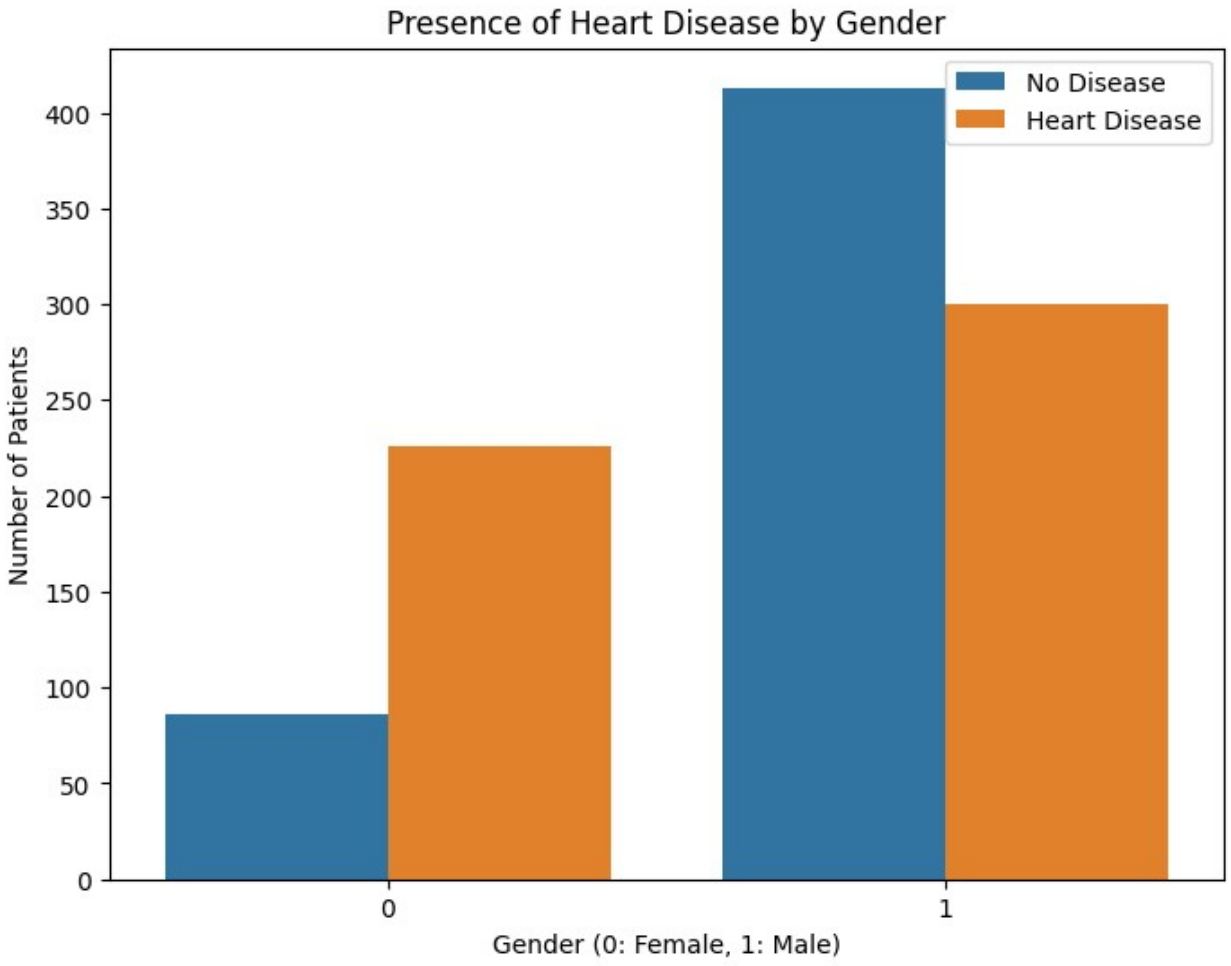
```
# Set the figure size
plt.figure(figsize=(10, 7))

# Create a histogram using Seaborn
sns.histplot(df['age'], kde=True)

# Add title and labels
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Number of Patients')

# Show the plot
plt.show()
```



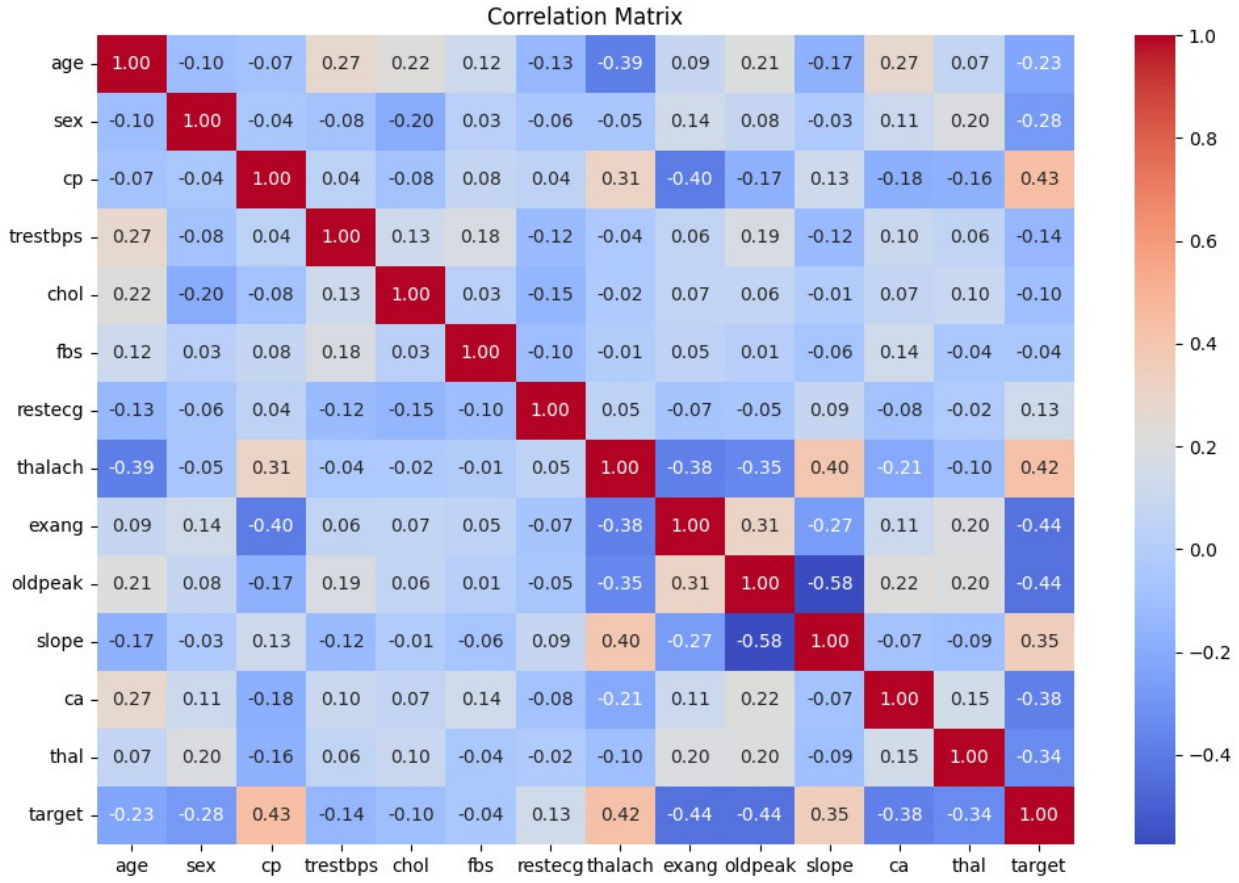

```
# Set the figure size
plt.figure(figsize=(10, 7))

# Create a histogram using Seaborn
sns.histplot(df['chol'], kde=True)

# Add title and labels
plt.title('Distribution of Cholesterol Level')
plt.xlabel('Cholesterol Level')
plt.ylabel('Number of Patients')

# Show the plot
plt.show()
```



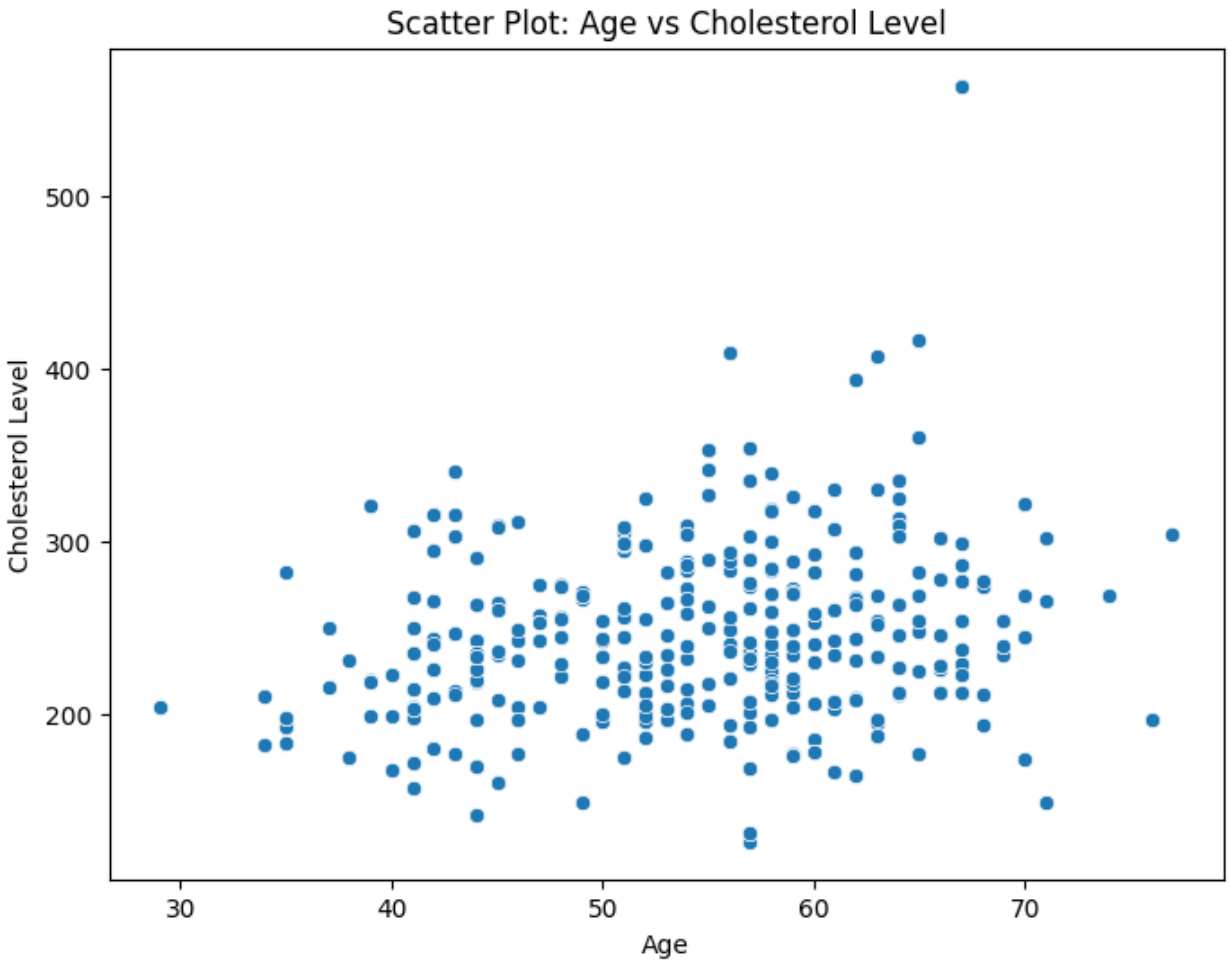

```
# Set the figure size
plt.figure(figsize=(8, 6))

# Create a countplot using Seaborn
sns.countplot(x='sex', data=df, hue='target')

# Add title and labels
plt.title('Presence of Heart Disease by Gender')
plt.xlabel('Gender (0: Female, 1: Male)')
plt.ylabel('Number of Patients')
plt.legend(['No Disease', 'Heart Disease'])

# Show the plot
plt.show()
```



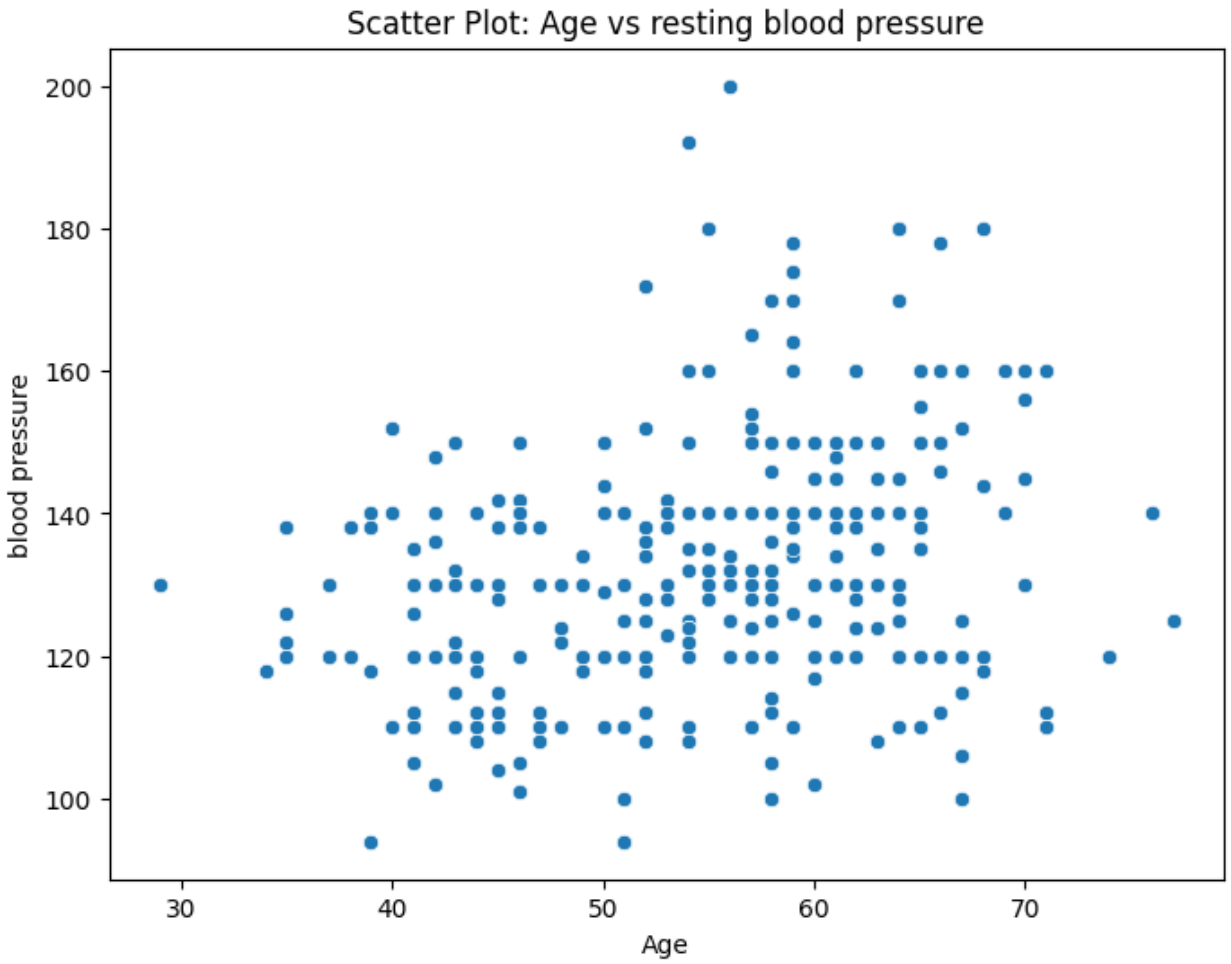

```
# Set the figure size
plt.figure(figsize=(12, 8))

# Create a heatmap using Seaborn
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')

# Add title
plt.title('Correlation Matrix')

# Show the plot
plt.show()
```




```
# Set the figure size
plt.figure(figsize=(8, 6))

# Create a scatter plot using Seaborn
sns.scatterplot(x='age', y='chol', data=df)

# Add title and labels
plt.title('Scatter Plot: Age vs Cholesterol Level')
plt.xlabel('Age')
plt.ylabel('Cholesterol Level')

# Show the plot
plt.show()
```




```
# Set the figure size
plt.figure(figsize=(8, 6))

# Create a scatter plot using Seaborn
sns.scatterplot(x='age', y='trestbps', data=df)

# Add title and labels
plt.title('Scatter Plot: Age vs resting blood pressure')
plt.xlabel('Age')
plt.ylabel('blood pressure')

# Show the plot
plt.show()
```



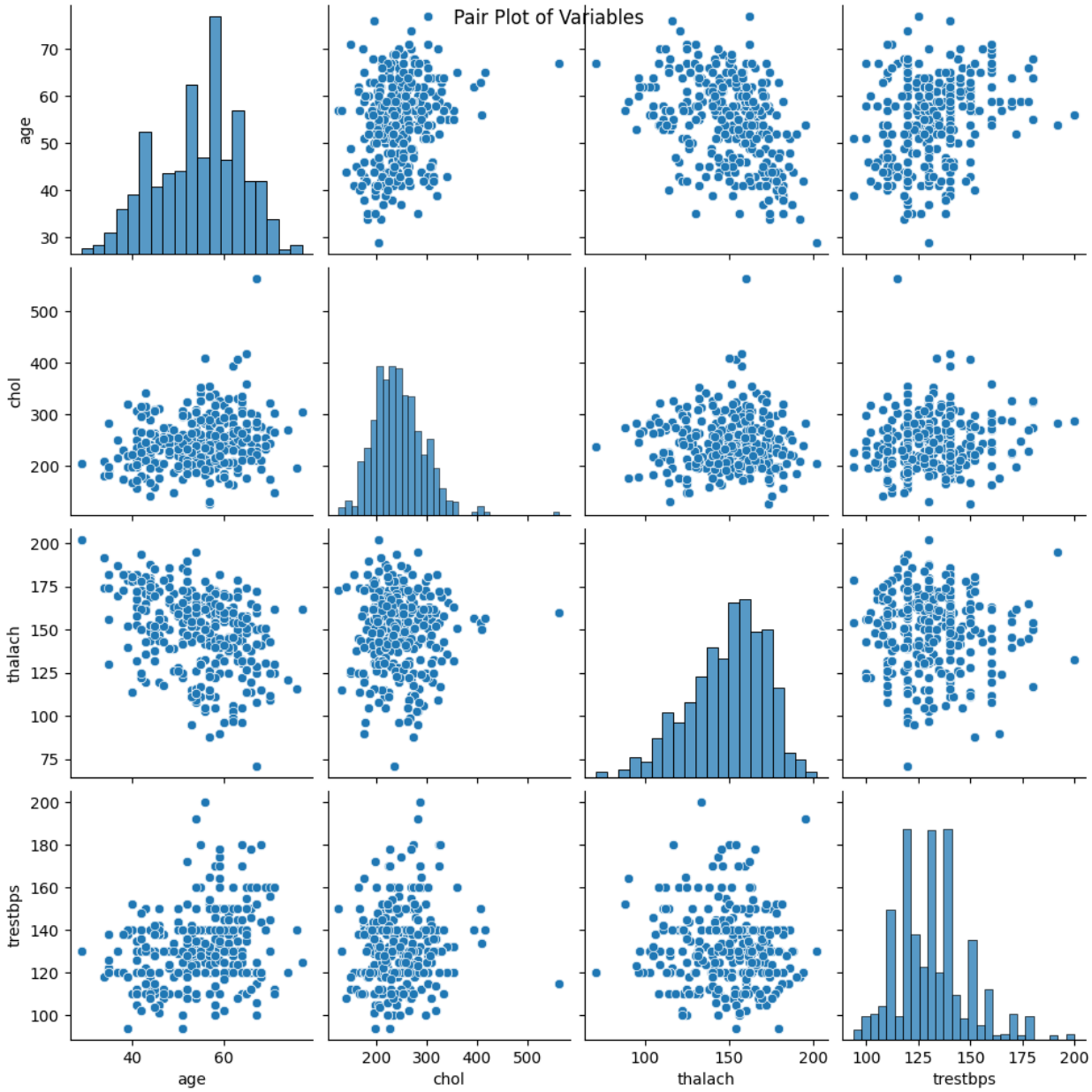

```
df.describe()
```

	age	sex	cp	trestbps	chol
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000
mean	54.434146	0.695610	0.942439	131.611707	246.00000
std	9.072290	0.460373	1.029641	17.516718	51.59251
min	29.000000	0.000000	0.000000	94.000000	126.00000
25%	48.000000	0.000000	0.000000	120.000000	211.00000
50%	56.000000	1.000000	1.000000	130.000000	240.00000
75%	61.000000	1.000000	2.000000	140.000000	275.00000
max	77.000000	1.000000	3.000000	200.000000	564.00000

```
                fbs      restecg      thalach        exang      oldpeak
\
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000

mean      0.149268     0.529756   149.114146     0.336585     1.071512

std       0.356527     0.527878    23.005724     0.472772     1.175053

min       0.000000     0.000000    71.000000     0.000000     0.000000

25%       0.000000     0.000000   132.000000     0.000000     0.000000

50%       0.000000     1.000000   152.000000     0.000000     0.800000

75%       0.000000     1.000000   166.000000     1.000000     1.800000

max       1.000000     2.000000   202.000000     1.000000     6.200000


              slope           ca         thal       target
count  1025.000000  1025.000000  1025.000000  1025.000000
mean      1.385366     0.754146     2.323902     0.513171
std       0.617755     1.030798     0.620660     0.500070
min       0.000000     0.000000     0.000000     0.000000
25%       1.000000     0.000000     2.000000     0.000000
50%       1.000000     0.000000     2.000000     1.000000
75%       2.000000     1.000000     3.000000     1.000000
max       2.000000     4.000000     3.000000     1.000000
```

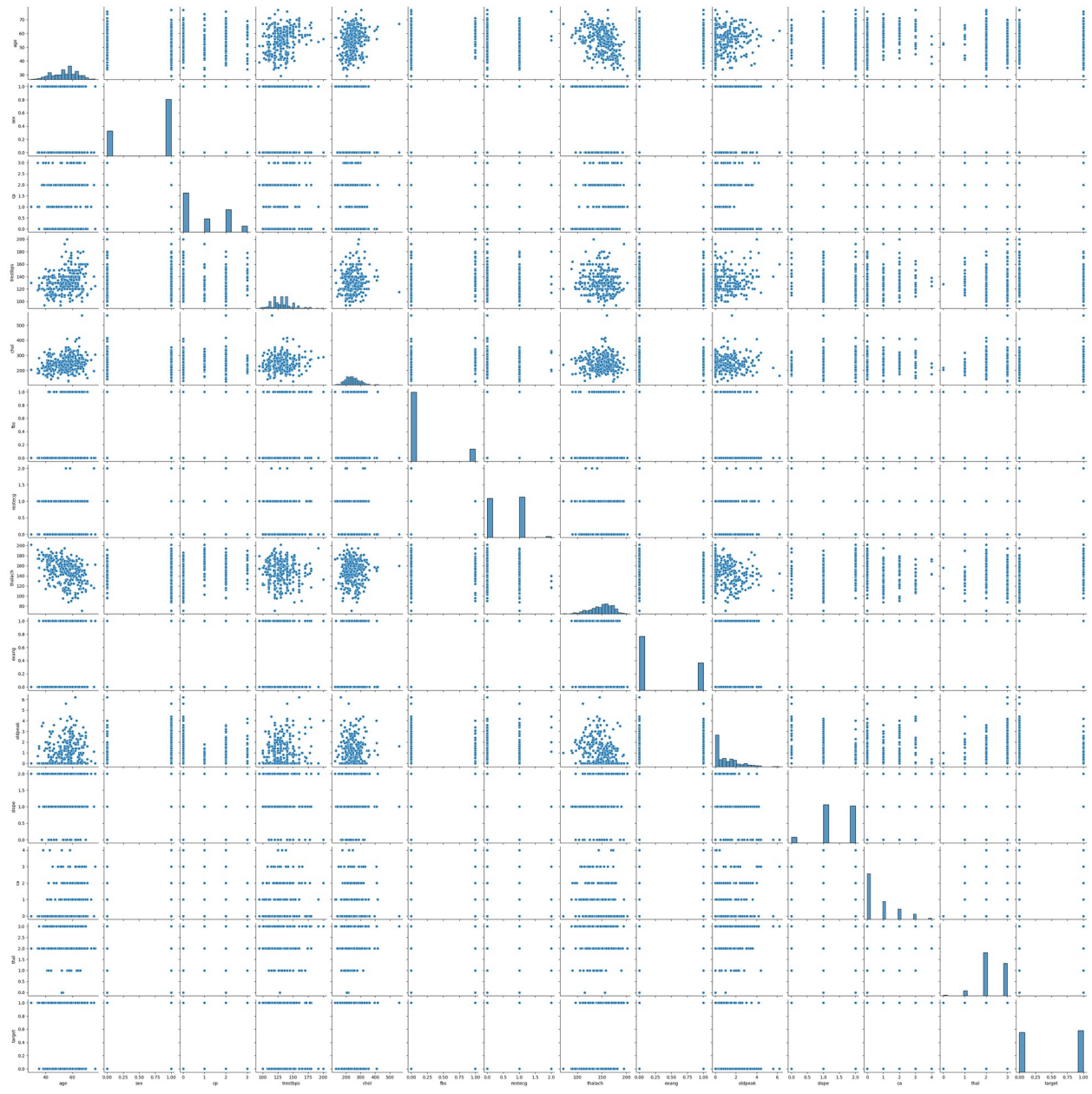
```python
# Set the figure size
plt.figure(figsize=(8, 6))

# Create a scatter plot using Seaborn
sns.scatterplot(x='age', y='thalach', data=df)

# Add title and labels
plt.title('Scatter Plot: Age vs maximum heart rate achieved')
plt.xlabel('Age')
plt.ylabel('maximum heart rate achieved')

# Show the plot
plt.show()
```


```
# Create a pair plot using Seaborn
sns.pairplot(df[['age', 'chol', 'thalach', 'trestbps']])

# Add a title
plt.suptitle('Pair Plot of Variables')

# Show the plot
plt.show()
```




```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x223bd8dd4d0>
```

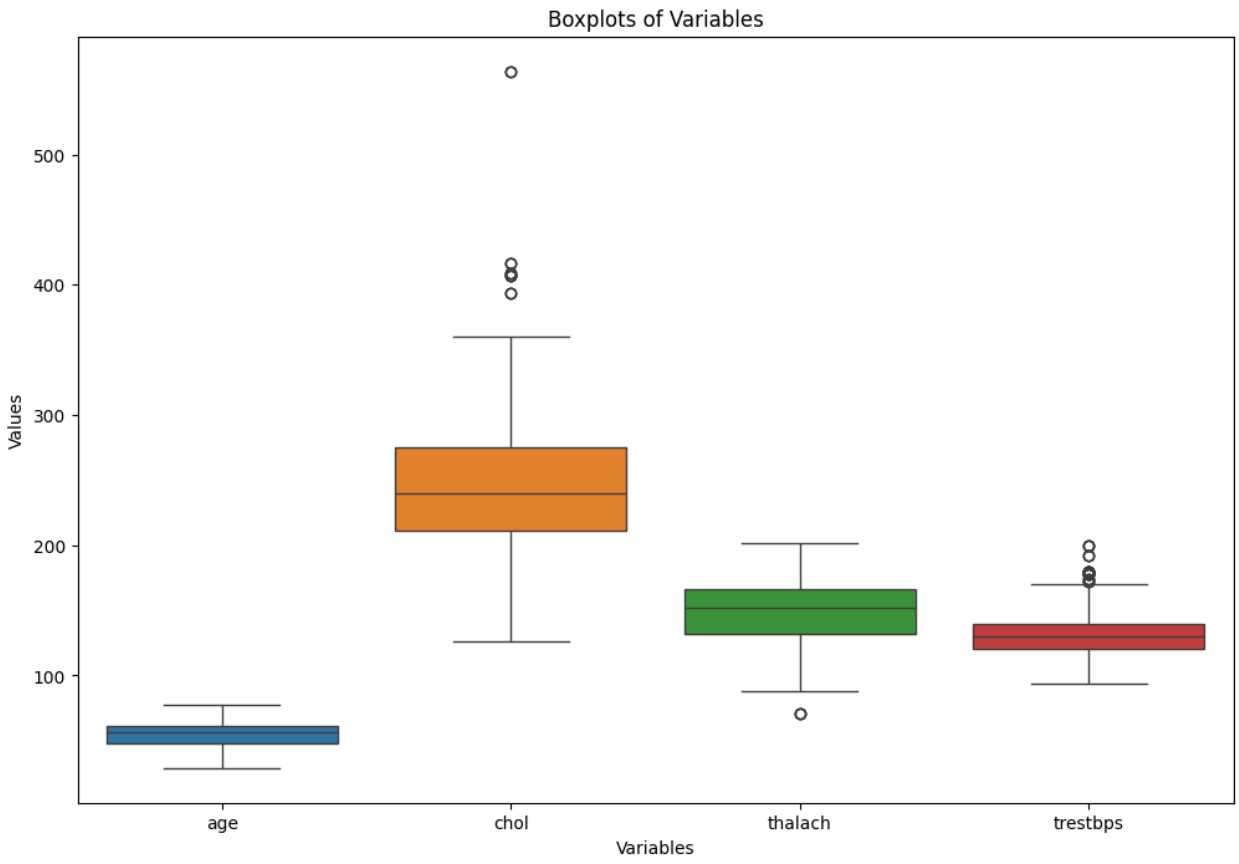


Data Preparation

Extra Work

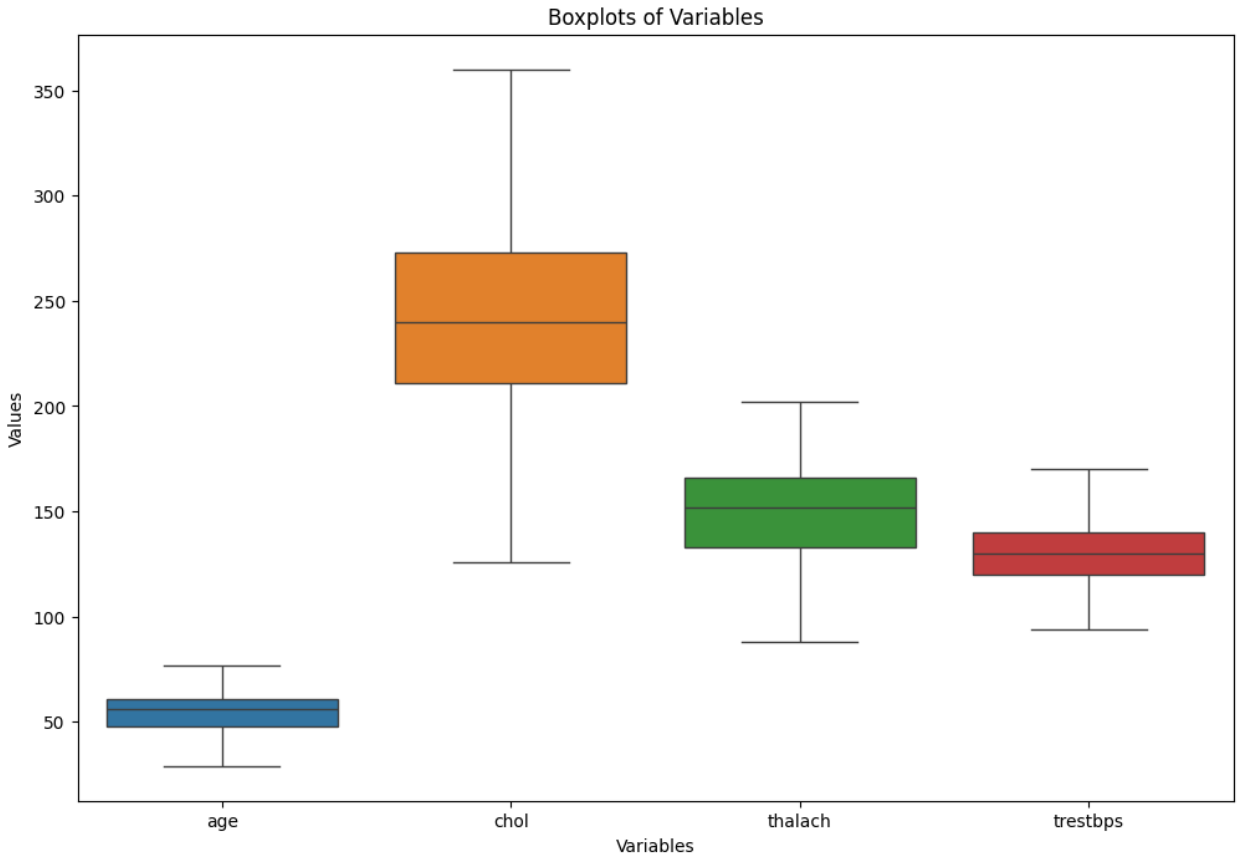
```
# Set the figure size
plt.figure(figsize=(12, 8))

# Create boxplots using Seaborn
sns.boxplot(data=df[['age', 'chol', 'thalach', 'trestbps']])

# Add title and labels
plt.title('Boxplots of Variables')
plt.xlabel('Variables')
```

```
plt.ylabel('Values')

# Show the plot
plt.show()
```




```
for col in df.columns:
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    df[col] = np.where((df[col] < lower_bound) | (df[col] >
upper_bound), df[col].mean(), df[col])

plt.figure(figsize=(12, 8))

# Create boxplots using Seaborn
sns.boxplot(df[['age', 'chol', 'thalach', 'trestbps']])

# Add title and labels
plt.title('Boxplots of Variables')
plt.xlabel('Variables')
plt.ylabel('Values')
```

```
# Show the plot
plt.show()
```




```
# selects columns with data types 'object', which typically represents
categorical variables.
# The tolist() method converts the column names to a list
categorical_cols =
df.select_dtypes(include=['object']).columns.tolist()

#  selects columns with data types 'int' and 'float', which represent
numerical variables.
numerical_cols = df.select_dtypes(include=['int',
'float']).columns.tolist()

# print the list of categorical column names.
print("Categorical column:", categorical_cols)

#  print the list of numerical column names.
print("Numerical column:", numerical_cols)
```

```
Categorical column: []
Numerical column: ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs',
```

```
'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal',
'target']

categorical_cols = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope',
'ca', 'thal']
numerical_cols = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

# ML libraries for data processing
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split

df_processed = df.copy()

scaler = StandardScaler()
df_processed[numerical_cols] =
scaler.fit_transform(df_processed[numerical_cols])

label_encoder = LabelEncoder()
df_processed[categorical_cols] =
df_processed[categorical_cols].apply(lambda col:
label_encoder.fit_transform(col))
```

Model Buliding

```
features = df_processed.drop('target', axis=1)
target = df_processed['target']

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42)

# X_train.shape, X_test.shape, y_train.shape, y_test.shape
print(f'The shape of training features is: {X_train.shape}')
print(f'The shape of training target is: {y_train.shape}')

print(f'The shape of testing features is: {X_test.shape}')
print(f'The shape of testing target is: {y_test.shape}')

The shape of training features is: (820, 13)
The shape of training target is: (820,)
The shape of testing features is: (205, 13)
The shape of testing target is: (205,)

from sklearn.neighbors import KNeighborsClassifier

# Instantiation of the 5-NN classifier
knn = KNeighborsClassifier(n_neighbors=5)

# Training the classifier on the training data
knn.fit(X_train, y_train)
```

```
# Prediction on the test data
y_pred = knn.predict(X_test)
```

Model Evaluation

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

# Calculation of accuracy
accuracy = accuracy_score(y_test, y_pred)

# Calculation of precision
precision = precision_score(y_test, y_pred)

# Calculation of recall
recall = recall_score(y_test, y_pred)

# Calculation of F1 score
f1 = f1_score(y_test, y_pred)

# Displaying the results
print("Accuracy: {:.2f}%".format(accuracy * 100))
print("Precision: {:.2f}%".format(precision * 100))
print("Recall: {:.2f}%".format(recall * 100))
print("F1 Score: {:.2f}%".format(f1 * 100))
```

```
Accuracy: 82.93%
Precision: 80.91%
Recall: 86.41%
F1 Score: 83.57%
```