□ Heart Disease Analysis

📊 Data Exploration & Insights

🔍 Objectives:

- Understand the key factors influencing heart disease.
- Perform Exploratory Data Analysis (EDA) on patient data.
- Build predictive models for diagnosis.

📌 **Dataset Overview:** The dataset consists of various health metrics like age, cholesterol levels, blood pressure, etc.

🛠 **Tools & Libraries:**

- Pandas
- NumPy
- Matplotlib & Seaborn (for visualization)

🚀 **Let's dive into the analysis!**

In [5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as st
```

In [ ]:
```
```

In [6]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [7]:
```
%matplotlib inline
```

In [ ]:
```
```

In [8]:
```
df = pd.read_csv(r'E:\Data Sets -Data Frames\HealthCare\heart.csv')
```

In [9]:
```
df.head()
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	t
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	

Exploratory data Analysis

In [11]:
```
print('The shape of the data set ',df.shape)
```

```
The shape of the data set  (303, 14)
```

Summary Of the Data sets

In [13]:
```
print('The Summary of the above data set are given below :',df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
The Summary of the above data set are given below : None
```

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
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
```

In [15]:
```
#### Statistical Properties of DataSets
df.describe()
```

Out[15]:

	age	sex	cp	trestbps	chol	fbs	restecg
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000

In [16]:
```
# View columns nanes
df.columns
```

Out[16]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [17]:
```
# checking the number of uniques values in target variable
df['target'].nunique()
```

Out[17]:
```
2
```

In [18]:
```
#viewing the target variable
df['target'].unique()
```

Out[18]:
```
array([1, 0], dtype=int64)
```

In [19]:
```
#Frequency Distribution of target variables
df['target'].value_counts()
```

Out[19]:
```
target
1    165
0    138
Name: count, dtype: int64
```

Visualize the frequency of Distribution of Target variable ()

The above plots confirms the finding thata ---

there are 165 patients suffering from heart disease

there are 138 patient who do not have any heart disease

In [22]:
```
df.groupby('sex')['target'].value_counts()
```

Out[22]:
```
sex  target
0    1          72
     0          24
1    0         114
     1          93
Name: count, dtype: int64
```

In [23]:
```
df.sex.nunique() # ---- #for checking the number of variable in sex columns
```

Out[23]:
```
2
```

In [82]:
```
df.head()
```

Out[82]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	t
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	

In [109…
```
df['target'].nunique()
```

Out[109…
```
2
```

In [98]:
```
df.groupby('cp')['target'].value_counts()
```

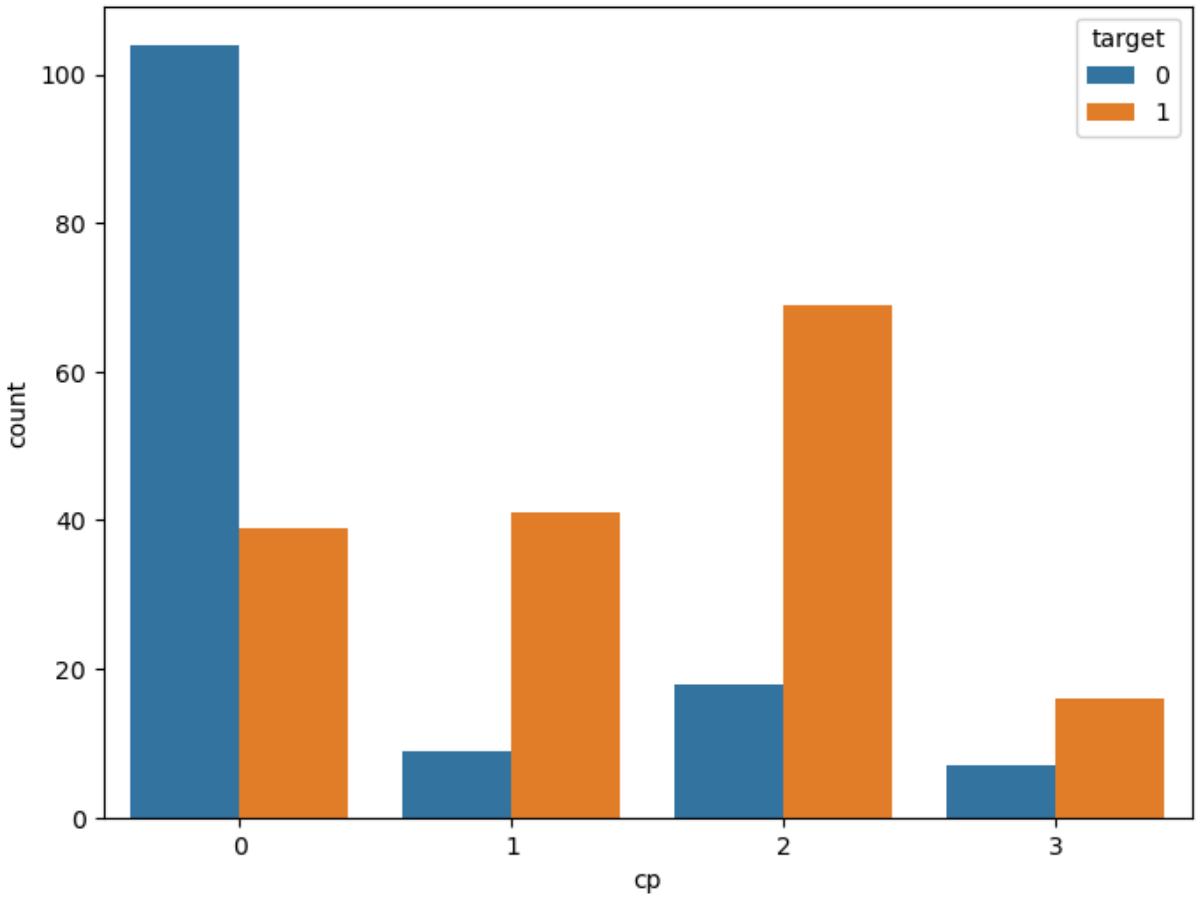
Out[98]:
```
cp  target
0   0         104
    1          39
1   1          41
    0           9
2   1          69
    0          18
3   1          16
    0           7
Name: count, dtype: int64
```

Observations form the data sets

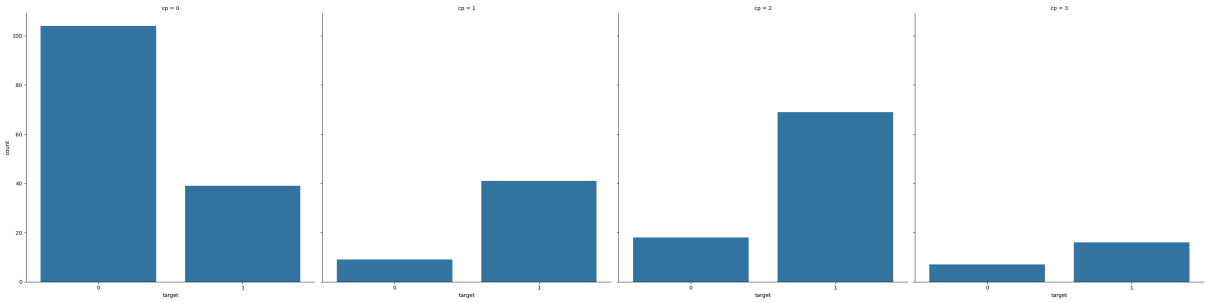
- `cp` variable contains four integer values 0, 1, 2 and 3.
- `target` variable contains two integer values 1 and 0 : (1 = Presence of heart disease; 0 = Absence of heart disease)
- So, the above analysis gives `target` variable values categorized into presence and absence of heart disease and groupby `cp` variable values.
- We can visualize this information below.

In [111…
```
### Visualization of the Graphs
```

In [113…
```
f , ax = plt.subplots(figsize=(8,6))
ax = sns.countplot(x='cp',hue='target',data=df)
plt.show()
```



In [125…
```
ax = sns.catplot(x="target", col="cp", data=df,
                 kind="count", height=8, aspect=1)
```



Eplore thalach Variable

- thalach stands for maximum heart rate achieved
- i will check the number of unique values in the thalach variables as follows

In [131…
```
df['thalach'].nunique()
```

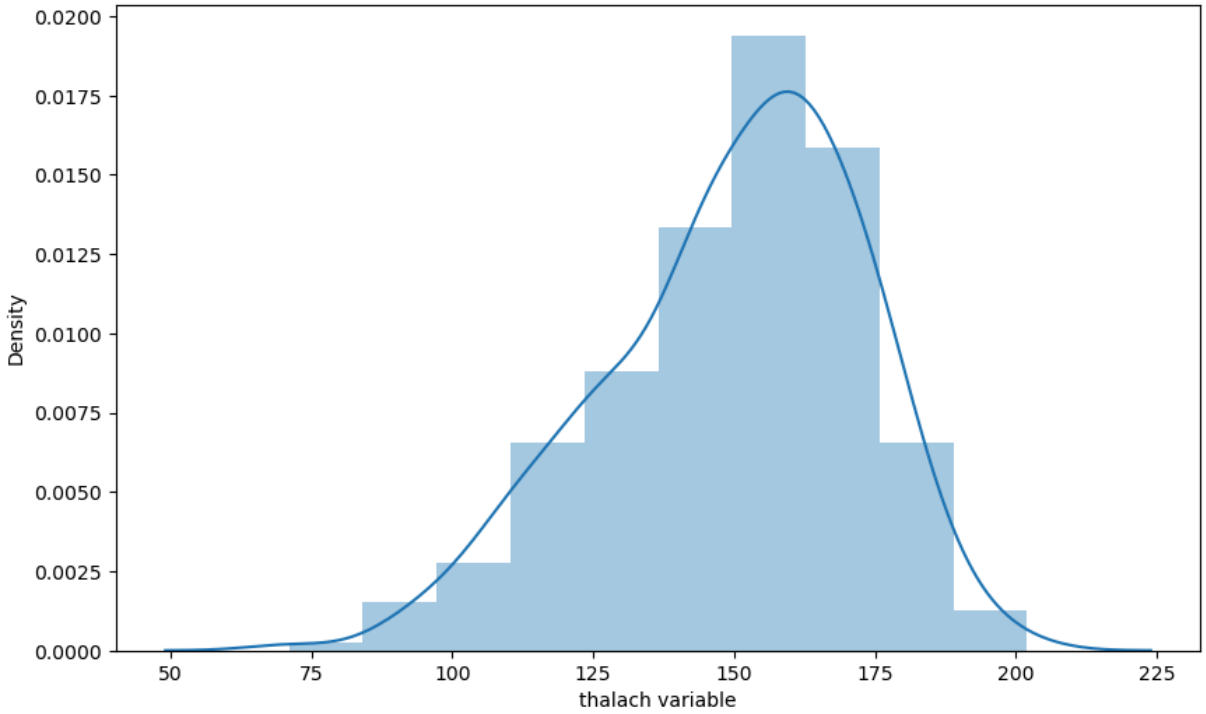
Out[131… 91

- so the numbers of uniues value in thalach varaiable is 92 .Hence it is numerical variable

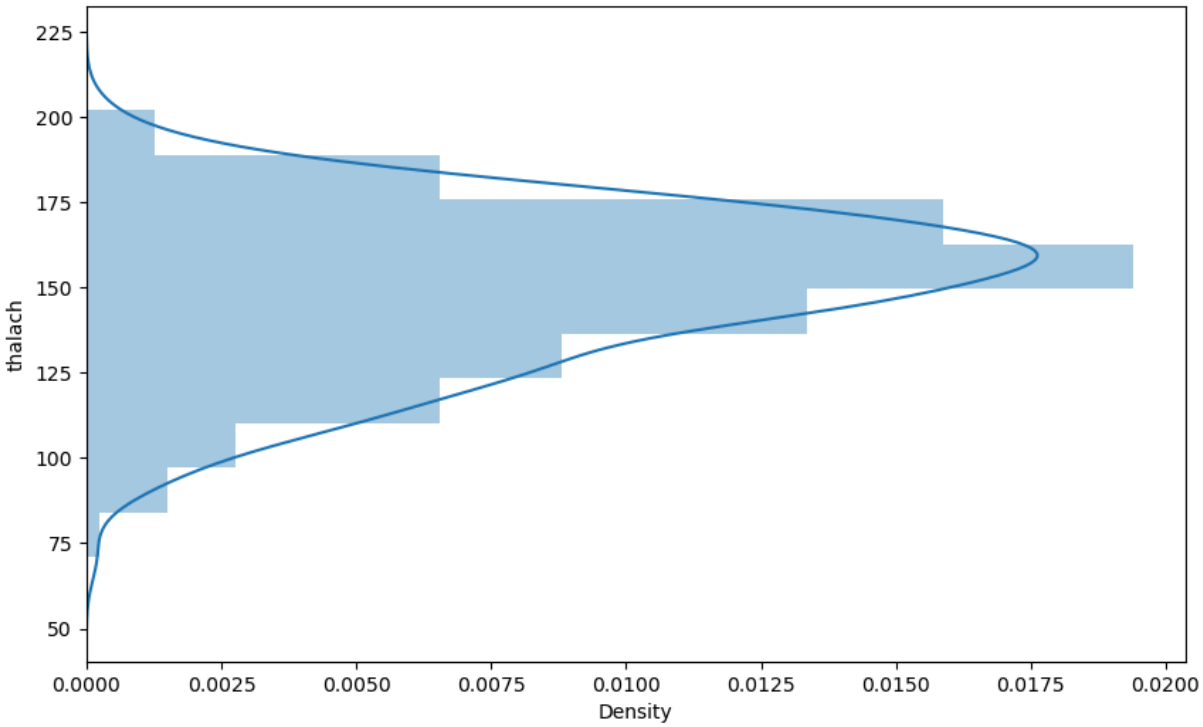
- it will visualize the frequency distribution as follow

- Visualize the frequency distribution distribution of thalach varaible

In [166…
```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
x = pd.Series(x, name="thalach variable")
ax = sns.distplot(x, bins=10)
plt.show()
```



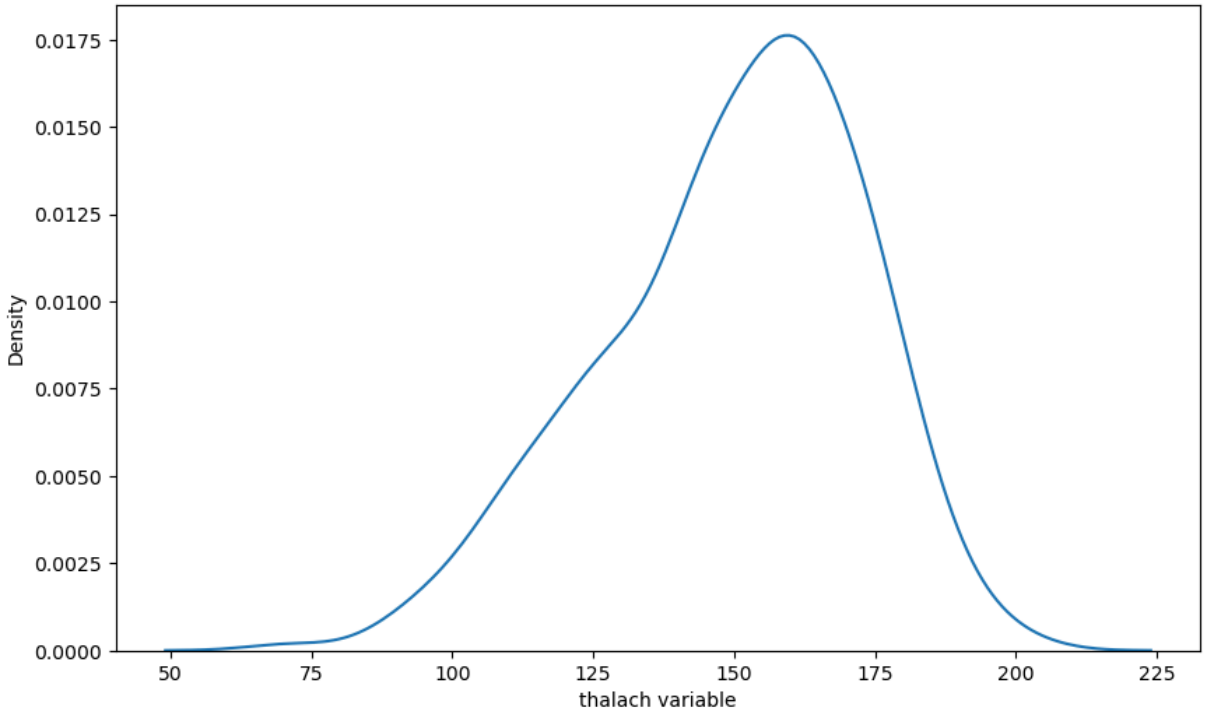
In [170…
```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
ax = sns.distplot(x, bins=10,vertical=True)
plt.show()
```



Seaborn Kernel Density Estimation (KDE Plots)

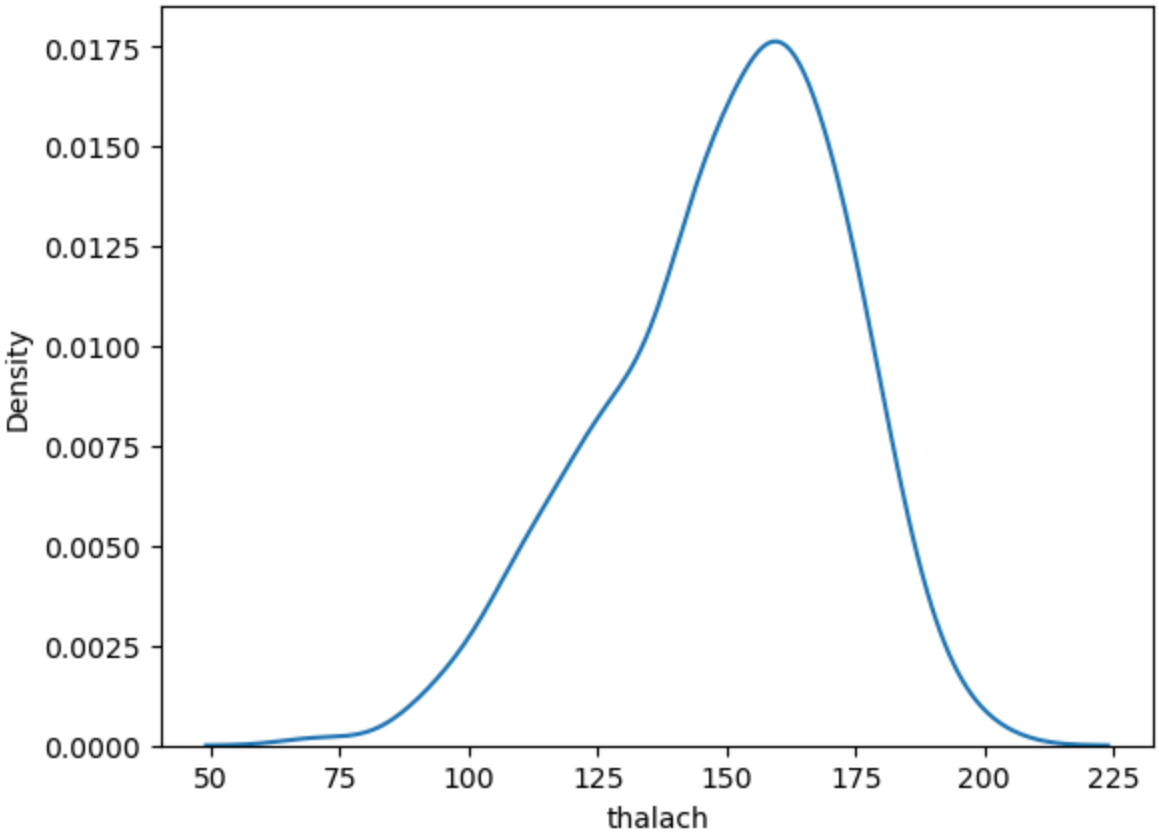
In [178…

```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
x = pd.Series(x, name="thalach variable")
ax = sns.kdeplot(x)
plt.show()
```



In [180…
```
# another way
sns.kdeplot(data= df , x = 'thalach'  )
```

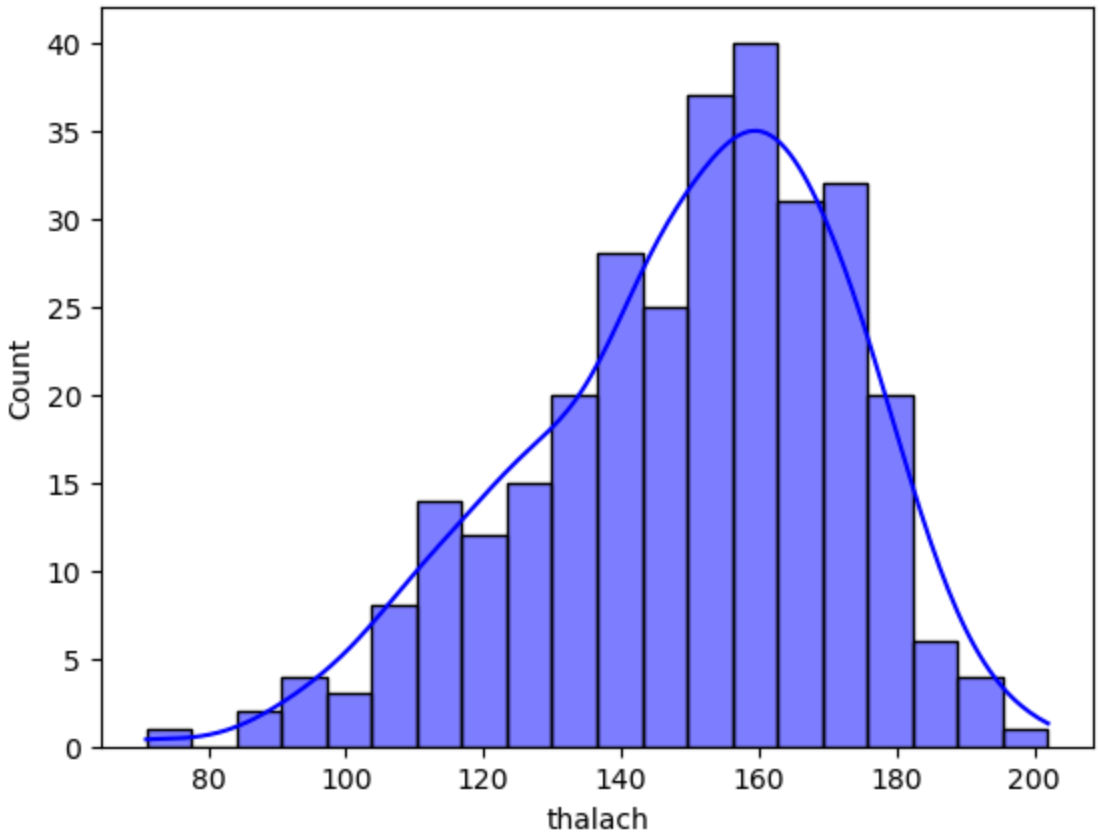
Out[180…
```
<Axes: xlabel='thalach', ylabel='Density'>
```



Histogram

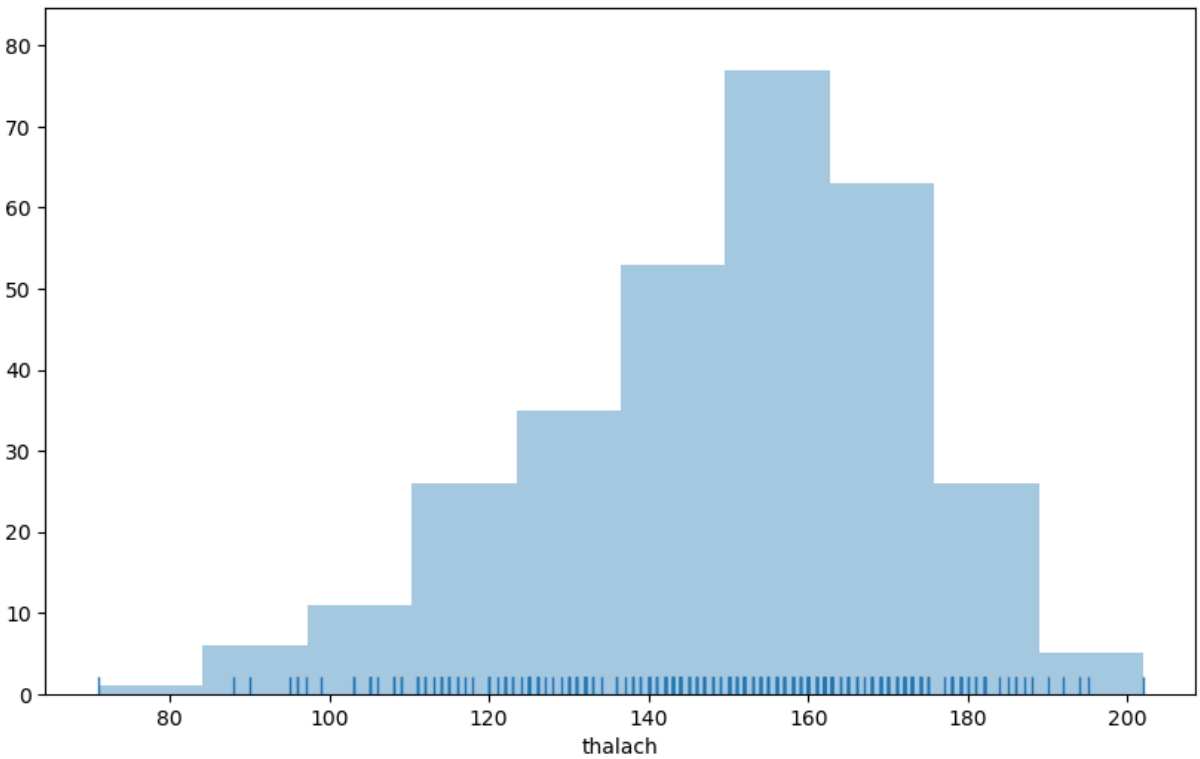
In [187…
```
sns.histplot(x='thalach',data=df,bins=20,kde=True,color='blue')
```

Out[187…
```
<Axes: xlabel='thalach', ylabel='Count'>
```



In [195…

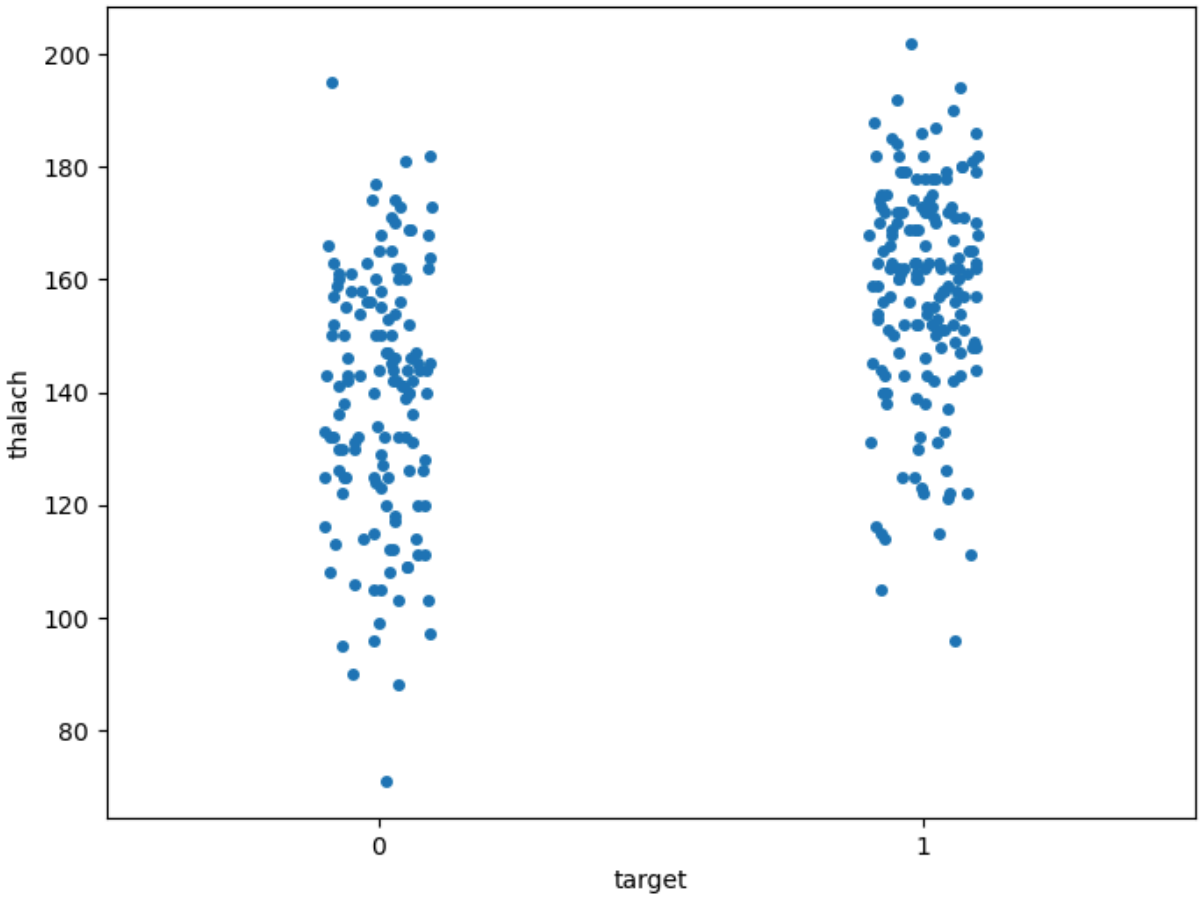
```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
ax = sns.distplot(x, kde=False, rug=True, bins=10)
plt.show()
```



Visualize frequency distribution of thalach variable wrt target

In [200…

```
f , ax = plt.subplots(figsize=(8,6))
sns.stripplot(x='target',y='thalach',data=df)
plt.show()
```

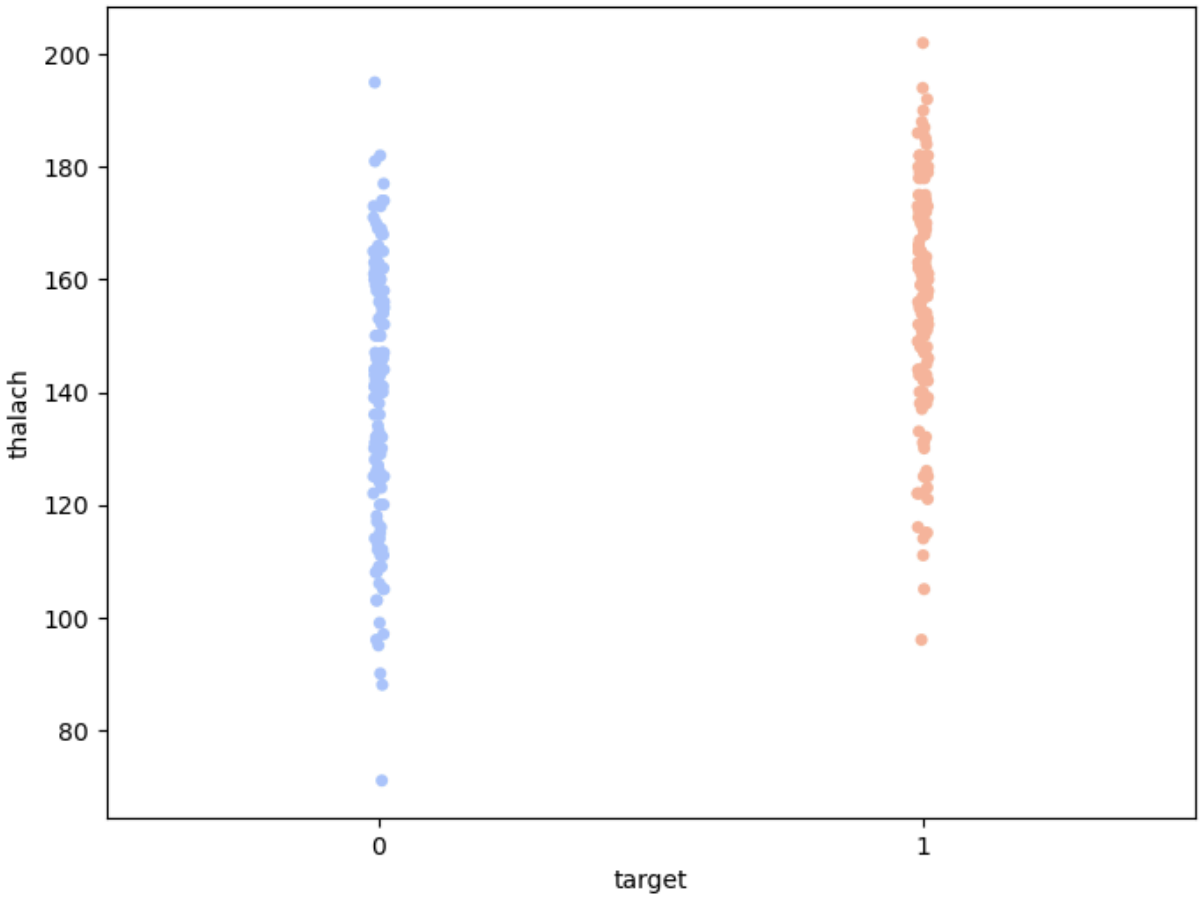


Interpretation

- We can see that those people suffering from heart disease (target = 1) have relatively higher heart rate (thalach) as compared to people who are not suffering from heart disease (target = 0).

In [205…

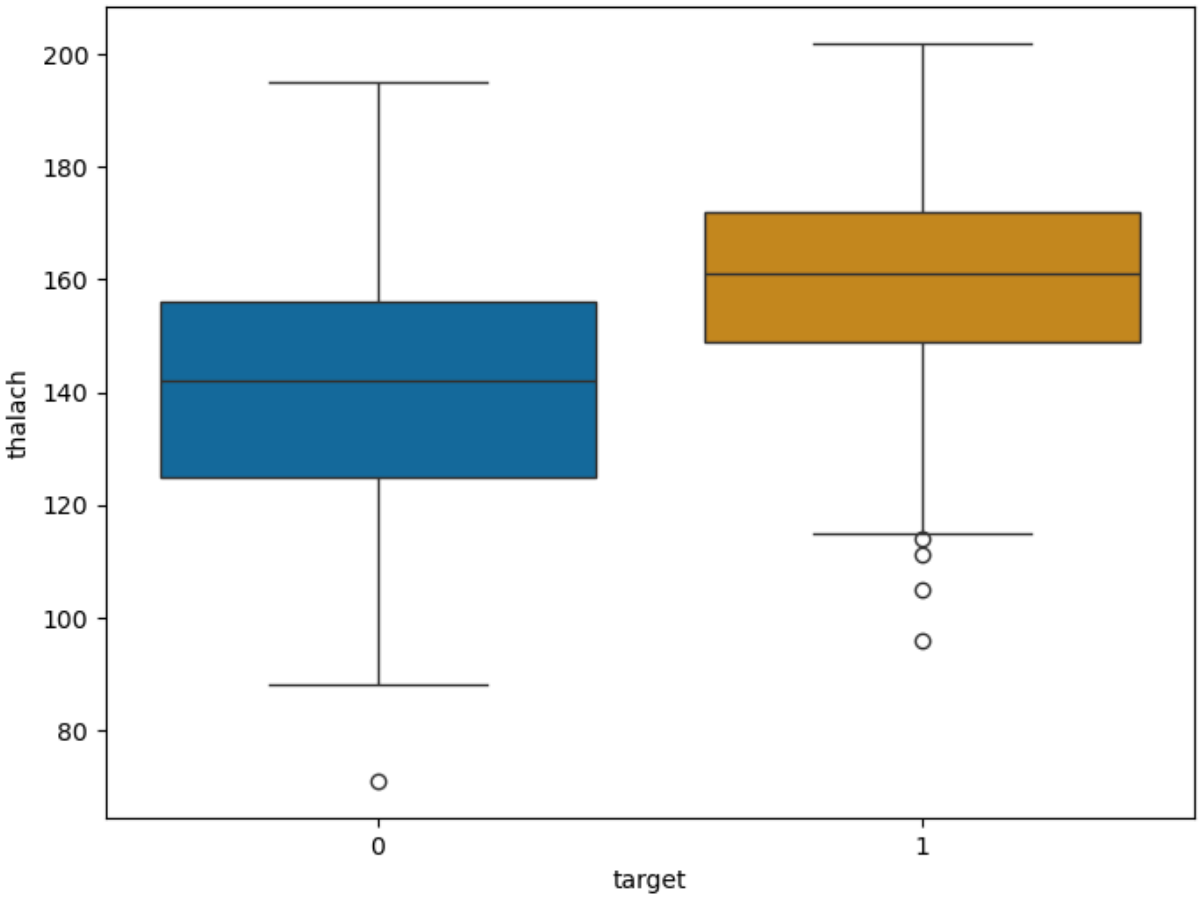
```
f ,ax = plt.subplots(figsize=(8,6))
sns.stripplot(data = df , x = 'target', y = 'thalach',jitter=0.01,palette='coolwarm
plt.show()
```



Visualize the distributions of thalach varaible wrt target with boxplots

In [218…

```
f , ax = plt.subplots(figsize=(8,6))
sns.boxplot(x = 'target',y='thalach',data=df,palette='colorblind')
plt.show()
```



Interpretation

The above boxplot confirms our finding that people suffering from heart disease (target = 1) have relatively higher heart rate (thalach) as compared to people who are not suffering from heart disease (target = 0).

Findings of Bivariate Analysis

Findings of Bivariate Analysis are as follows –

- There is no variable which has strong positive correlation with `target` variable.
- There is no variable which has strong negative correlation with `target` variable.
- There is no correlation between `target` and `fbs` .
- The `cp` and `thalach` variables are mildly positively correlated with `target` variable.
- We can see that the `thalach` variable is slightly negatively skewed.
- The people suffering from heart disease (target = 1) have relatively higher heart rate (thalach) as compared to people who are not suffering from heart disease (target = 0).

- The people suffering from heart disease (target = 1) have relatively higher heart rate (thalach) as compared to people who are not suffering from heart disease (target = 0).

Multivariate analysis

- The objective of the multivariate analysis is to discover patterns and relationships in the dataset.

Discover the patterns and relationships

In [256…

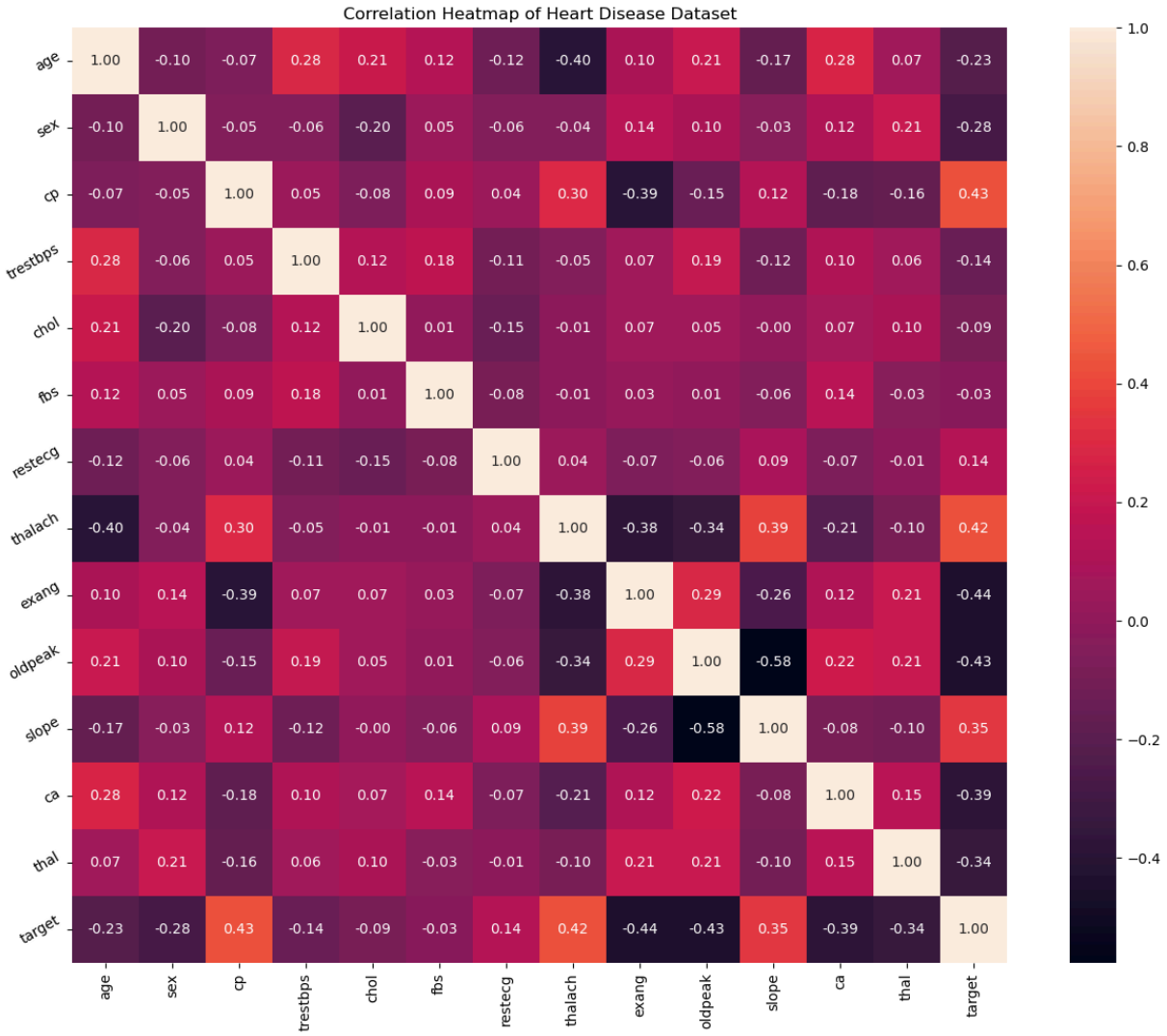
```
plt.figure(figsize=(16,12))  # Sets the figure size (16x12 inches)

plt.title('Correlation Heatmap of Heart Disease Dataset')  # Sets the title

a = sns.heatmap(df.corr(), square=True, annot=True,  # Creates the heatmap
                fmt='.2f', linecolor='white')

a.set_xticklabels(a.get_xticklabels(), rotation=90)  # Rotates x-axis labels
a.set_yticklabels(a.get_yticklabels(), rotation=30)  # Rotates y-axis labels

plt.show()  # Displays the heatmap
```



In [258…

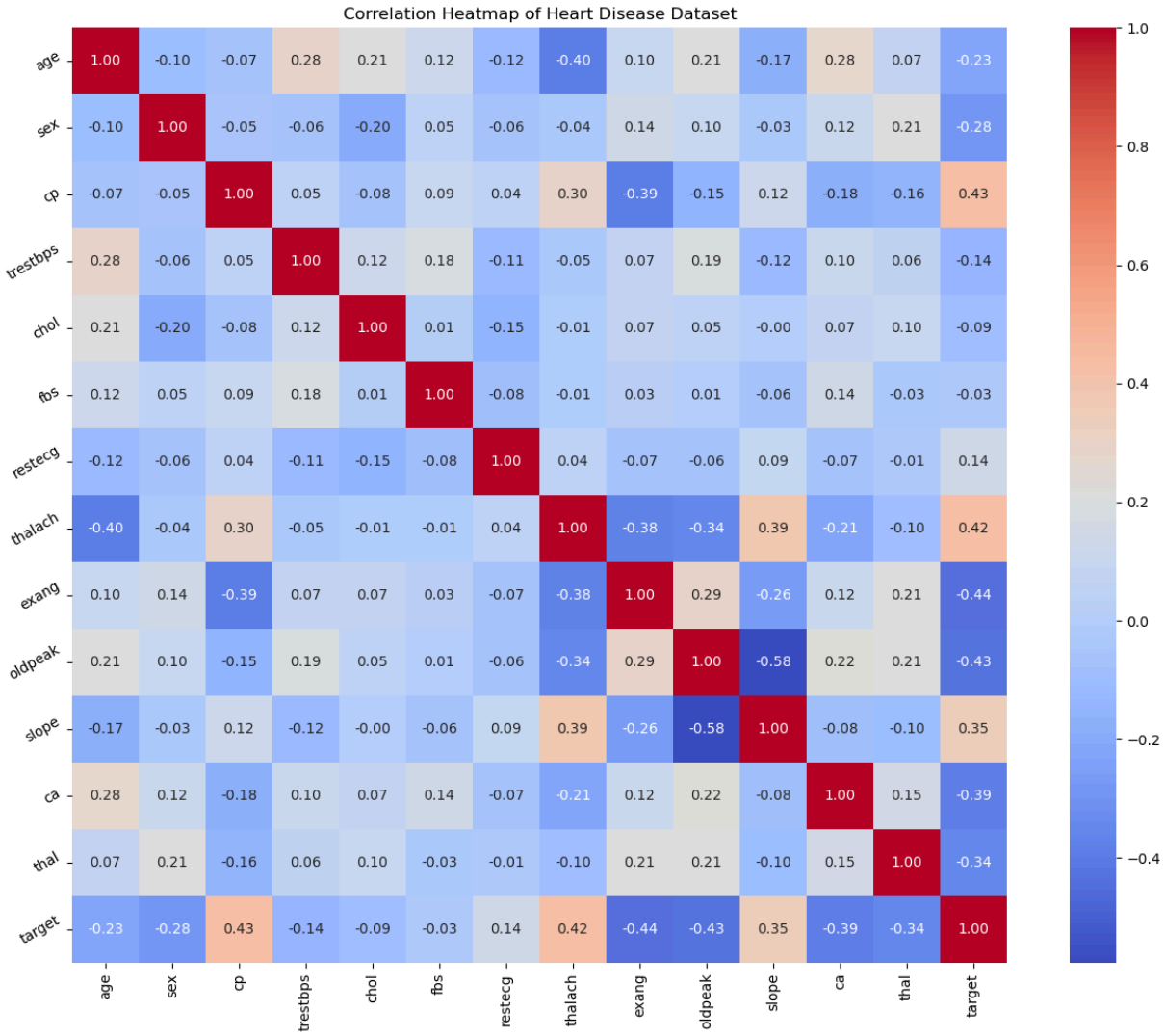
```
plt.figure(figsize=(16,12))  # Sets the figure size (16x12 inches)

plt.title('Correlation Heatmap of Heart Disease Dataset')  # Sets the title

a = sns.heatmap(df.corr(), square=True, annot=True,  # Creates the heatmap
                fmt='.2f', linecolor='white',cmap='coolwarm')

a.set_xticklabels(a.get_xticklabels(), rotation=90)  # Rotates x-axis labels
a.set_yticklabels(a.get_yticklabels(), rotation=30)  # Rotates y-axis labels

plt.show()  # Displays the heatmap
```



Interpretation

From the above correlation heat map, we can conclude that :-

- `target` and `cp` variable are mildly positively correlated (correlation coefficient = 0.43).
- `target` and `thalach` variable are also mildly positively correlated (correlation coefficient = 0.42).
- `target` and `slope` variable are weakly positively correlated (correlation coefficient = 0.35).
- `target` and `exang` variable are mildly negatively correlated (correlation coefficient = -0.44).
- `target` and `oldpeak` variable are also mildly negatively correlated (correlation coefficient = -0.43).

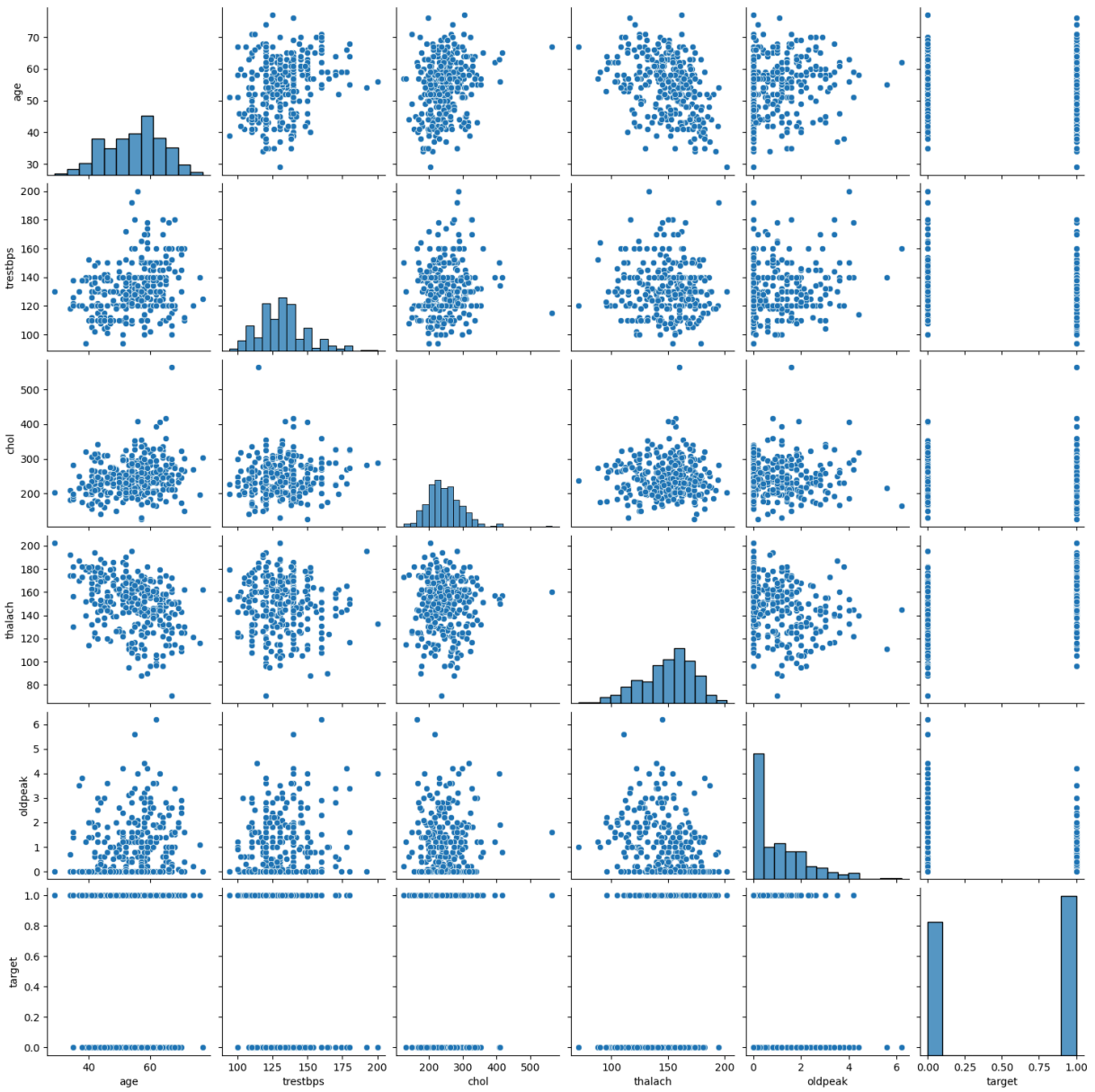
- `target` and `ca` variable are weakly negatively correlated (correlation coefficient = -0.39).
- `target` and `thal` variable are also waekly negatively correlated (correlation coefficient = -0.34).

Pair Plot

- Showing the realtionship between multiple varaible

In [269…

```
num_var =['age', 'trestbps', 'chol', 'thalach', 'oldpeak','target' ]
sns.pairplot(df[num_var],kind='scatter',diag_kind='hist')
plt.show()
```



Observation from the aboe pairplot

- I have defined a variable `num_var`. Here `age`, `trestbps`, `chol`, `thalach` and `oldpeak` are numerical variables and `target` is the categorical variable.
- So, I wll check relationships between these variables.

Analysis of age and other varaiables

Check the nmber of unique variable in age variable

In [275…
```
df['age'].nunique()
```

Out[275…
```
41
```

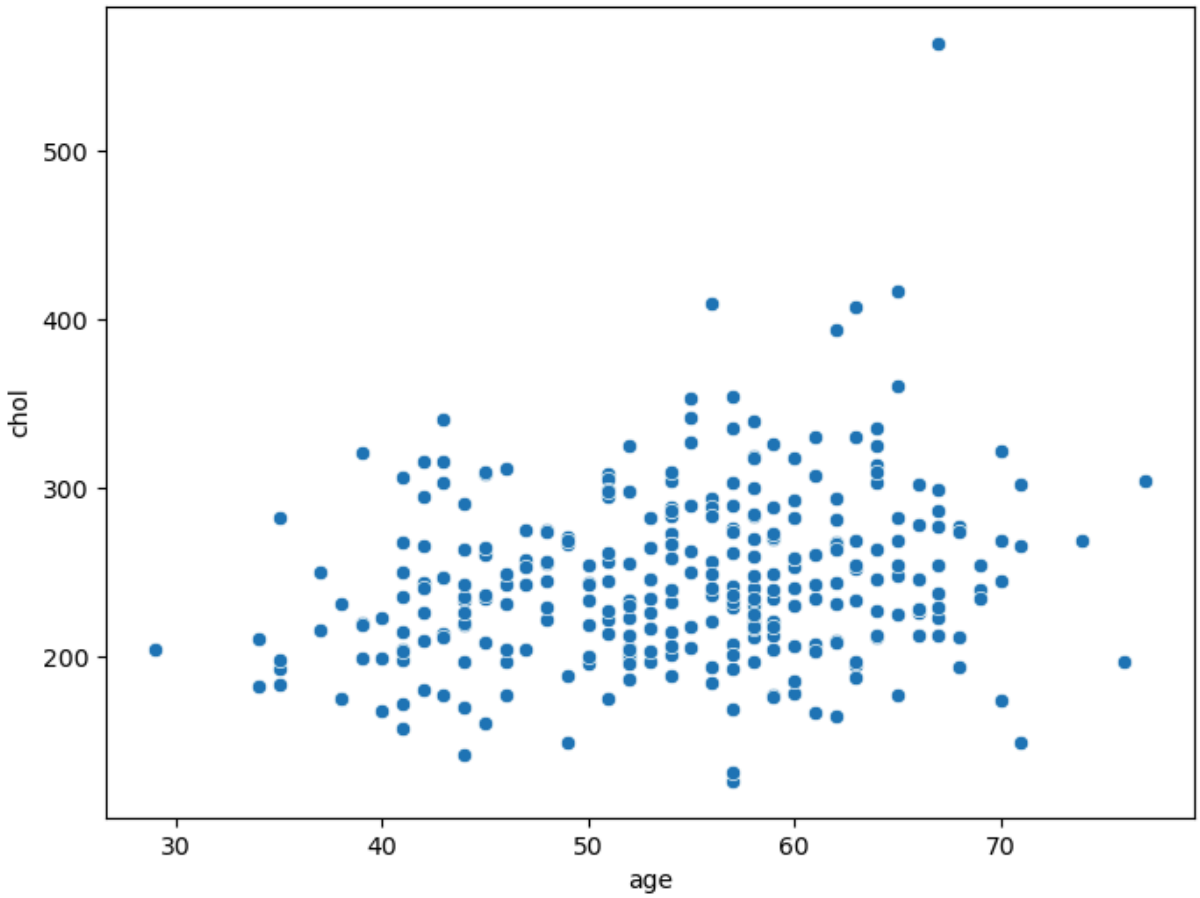
Viewing the statistical Summary of age variables

In [278…
```
df['age'].describe()
```

Out[278…
```
count    303.000000
mean      54.366337
std        9.082101
min       29.000000
25%       47.500000
50%       55.000000
75%       61.000000
max       77.000000
Name: age, dtype: float64
```

Analze age and chol varaiable

In [285…
```
f ,ax = plt.subplots(figsize=(8,6))
ax = sns.scatterplot(x='age',y='chol',data=df)
plt.show()
```



Interpretation

In [ ]:
```
- The above plots confirms thata there is slighly positive correlation between age
```

10. Dealing with missing values

Back to Table of Contents

- In Pandas missing data is represented by two values:
  - **None**: None is a Python singleton object that is often used for missing data in Python code.
  - **NaN** : NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation.
- There are different methods in place on how to detect missing values.

Pandas isnull() and notnull() functions

- Pandas offers two functions to test for missing data - `isnull()` and `notnull()`. These are simple functions that return a boolean value indicating whether the passed in

argument value is in fact missing data.
- Below, I will list some useful commands to deal with missing values.

Useful commands to detect missing values

- **df.isnull()**

The above command checks whether each cell in a dataframe contains missing values or not. If the cell contains missing value, it returns True otherwise it returns False.

- **df.isnull().sum()**

The above command returns total number of missing values in each column in the dataframe.

- **df.isnull().sum().sum()**

It returns total number of missing values in the dataframe.

- **df.isnull().mean()**

It returns percentage of missing values in each column in the dataframe.

- **df.isnull().any()**

It checks which column has null values and which has not. The columns which has null values returns TRUE and FALSE otherwise.

- **df.isnull().any().any()**

It returns a boolean value indicating whether the dataframe has missing values or not. If dataframe contains missing values it returns TRUE and FALSE otherwise.

- **df.isnull().values.any()**

It checks whether a particular column has missing values or not. If the column contains missing values, then it returns TRUE otherwise FALSE.

- **df.isnull().values.sum()**

It returns the total number of missing values in the dataframe.

In [291…
```
# Checking for the missing values
df.isnull().sum()
```

Out[291…
```
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
```

Observations

we can see that there is no missing value in the data sets

Assert Statements

- We must confirm that our dataset has no missing values.
- We can write an **assert statement** to verify this.
- We can use an assert statement to programmatically check that no missing, unexpected 0 or negative values are present.
- This gives us confidence that our code is running properly.
- **Assert statement** will return nothing if the value being tested is true and will throw an AssertionError if the value is false.
- **Asserts**
  - assert 1 == 1 (return Nothing if the value is True)
  - assert 1 == 2 (return AssertionError if the value is False)

In [303…
```
assert pd.notnull(df).all().all()
```

In [307…
```
# assert values are greater than or eqaul to 0
assert (df >= 0).all().all()
```

obseravtions

- The above two command does not throw any error . Hence it is confirmed that there are no missing or ngative values in the datasets
- All the values are greater than or equal to zero

outlier detections

- I will make boxplot to visualise outlier in the continuous numerical varaible :

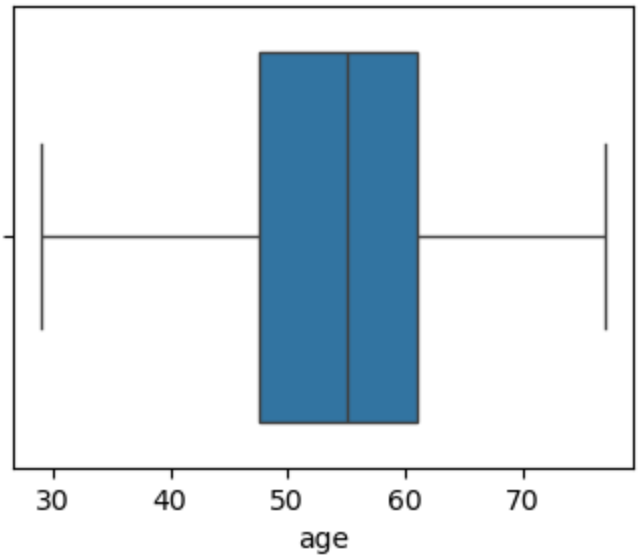
`age` , `trestbps` , `chol` , `thalach` and `oldpeak` variables.

In [312…
```
df['age'].describe()
```

Out[312…
```
count    303.000000
mean      54.366337
std        9.082101
min       29.000000
25%       47.500000
50%       55.000000
75%       61.000000
max       77.000000
Name: age, dtype: float64
```

boxplot of age varaiable

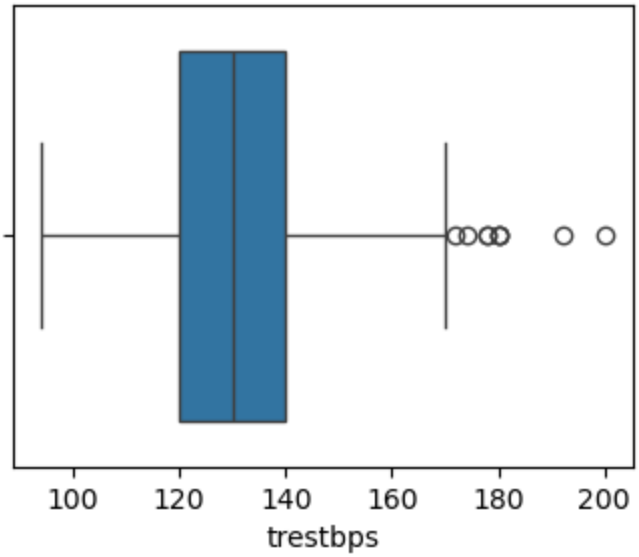
In [323…
```
f ,ax = plt.subplots(figsize=(4,3))
ax = sns.boxplot(x=df['age'])
plt.show()
```



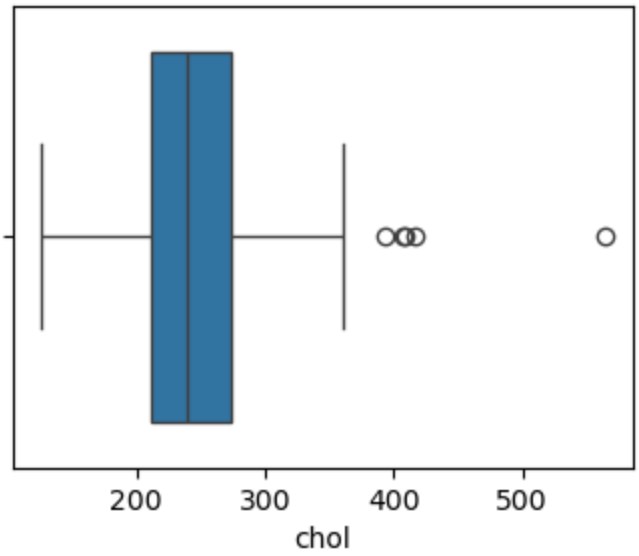
In [325…
```
# trestbps varaiable
df['trestbps'].describe()
```

Out[325…
```
count    303.000000
mean     131.623762
std       17.538143
min       94.000000
25%      120.000000
50%      130.000000
75%      140.000000
max      200.000000
Name: trestbps, dtype: float64
```

In [327…
```
f ,ax = plt.subplots(figsize=(4,3))
ax = sns.boxplot(x=df['trestbps'])
plt.show()
```

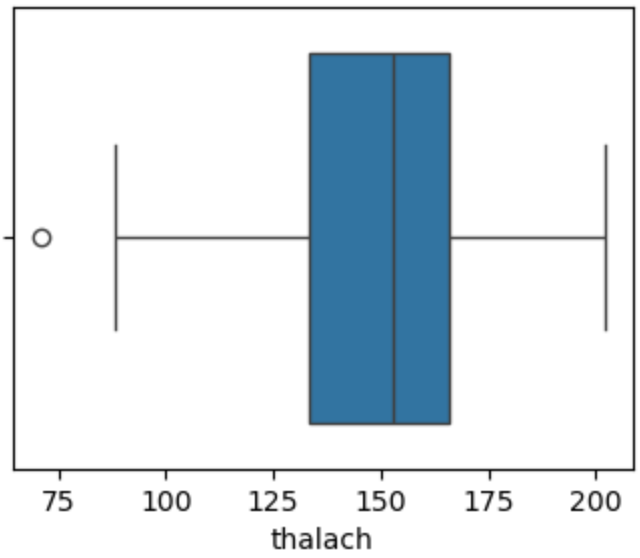


In [331…
```
# chol varaiable
f ,ax = plt.subplots(figsize=(4,3))
ax = sns.boxplot(x=df['chol'])
plt.show()
```



thalach varaiable

In [336…
```
f ,ax = plt.subplots(figsize=(4,3))
ax = sns.boxplot(x=df['thalach'])
plt.show()
```

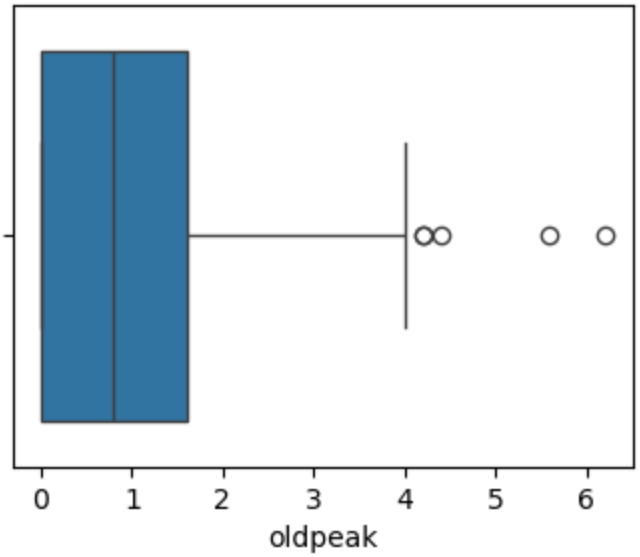


oldpeak varaiable

In [339…
```
df['oldpeak'].describe()
```

Out[339…
```
count    303.000000
mean       1.039604
std        1.161075
min        0.000000
25%        0.000000
50%        0.800000
75%        1.600000
max        6.200000
Name: oldpeak, dtype: float64
```

In [341…
```
f ,ax = plt.subplots(figsize=(4,3))
ax = sns.boxplot(x=df['oldpeak'])
plt.show()
```



Findings

- The `age` variable does not contain any outlier.
- `trestbps` variable contains outliers to the right side.
- `chol` variable also contains outliers to the right side.
- `thalach` variable contains a single outlier to the left side.
- `oldpeak` variable contains outliers to the right side.
- Those variables containing outliers needs further investigation.

Conculsions

Our EDA journey has come to an end.

In this kernel, we have explored the heart disease dataset. The feature variable of interest is `target` variable. We have analyzed it alone and check its interaction with other variables. We have also discussed how to detect missing data and `outliers`.

Thanks

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: