Comprehensive EDA on Heart Disease UCI Dataset -

Extensive Analysis + Visualization with Python

Heart disease or **Cardiovascular disease (CVD)** is a class of diseases that involve the heart or blood vessels. Cardiovascular diseases are the leading cause of death globally. This is true in all areas of the world except Africa. Together CVD resulted in 17.9 million deaths (32.1%) in 2015. Deaths, at a given age, from CVD are more common and have been increasing in much of the developing world, while rates have declined in most of the developed world since the 1970s.

Import libraries

In [3]:
```
import pandas as pd  #data processing
import numpy as np #linear algebra
```

In [105…
```
import os
```

In [107…
```
import seaborn as sns  #statistical graphics
import matplotlib.pyplot as plt  #Data Visualization
import scipy.stats as st  #Statistical Analysis
%matplotlib inline

sns.set(style='whitegrid')
```

In [109…
```
import warnings
warnings.filterwarnings('ignore')
```

I will import the dataset with the usual `pandas read_csv()` function which is used to import CSV (Comma Separated Value) files.

In [111…
```
df=pd.read_csv(r"C:\Users\kench\OneDrive\Desktop\My Folder\DSwP\05-03 _ seaborn,
```

Dataset description

- The dataset contains several columns which are as follows -
  - age : age in years
  - sex : (1 = male; 0 = female)
  - cp : chest pain type
  - trestbps : resting blood pressure (in mm Hg on admission to the hospital)
  - chol : serum cholestoral in mg/dl
  - fbs : (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
  - restecg : resting electrocardiographic results
  - thalach : maximum heart rate achieved
  - exang : exercise induced angina (1 = yes; 0 = no)

- oldpeak : ST depression induced by exercise relative to rest
- slope : the slope of the peak exercise ST segment
- ca : number of major vessels (0-3) colored by flourosopy
- thal : 3 = normal; 6 = fixed defect; 7 = reversable defect
- target : 1 or 0

In [114…
```
df
```

Out[114…

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tl
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	

303 rows × 14 columns

Check shape of the dataset

- It is a good idea to first check the shape of the dataset.

In [342…
```
print("The shape of the dataset:",df.shape)
```

```
The shape of the dataset: (303, 14)
```

preview dataset

In [118…
```
df.head()
```

Out[118…

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
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 [120…

```
df.tail()
```

Out[120…

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tl
298	57	0	0	140	241	0	1	123	1	0.2	1	0	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	

Summary of dataset

In [122…

```
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
```

Statistical properties of dataset

In [124…

```
df.describe()
```

Out[124...

	age	sex	cp	trestbps	chol	fbs	reste
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000(
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528(
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525ε
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000(
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000(
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000(
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000(
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000(

In [126...

```
df.dtypes
```

Out[126...

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [128...

```
df.columns
```

Out[128...

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

Check the number of unique values in `target` variable

In [130...

```
df.target.nunique()
```

Out[130...

```
2
```

View the unique values in `target` variable

In [132...

```
df.target.unique()
```

Out[132...

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

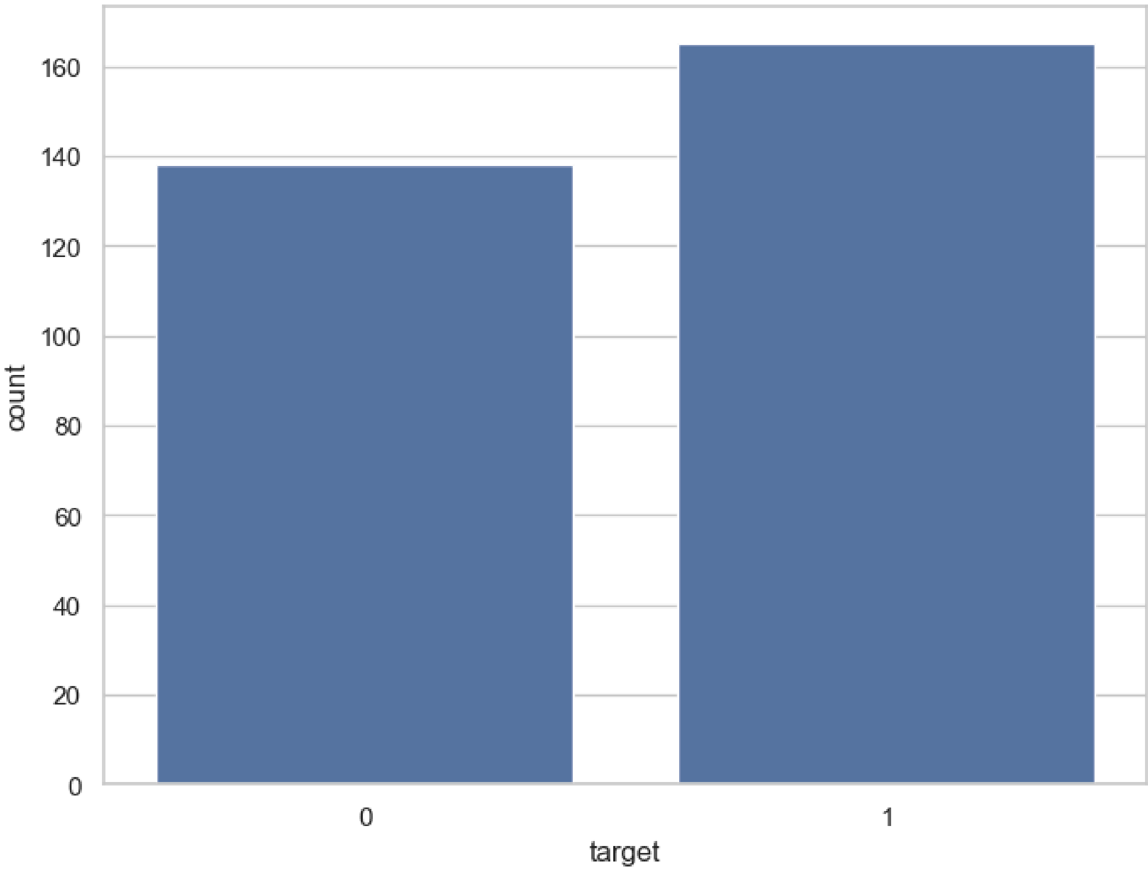
Frequency distribution of `target` variable

In [134...
```
df.target.value_counts()
```

Out[134...
```
target
1    165
0    138
Name: count, dtype: int64
```

Visualize frequency distribution of `target` variable

In [136...
```
f, ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="target",data=df)
plt.show()
```



The above plot confirms the findings that -

- There are 165 patients suffering from heart disease, and
- There are 138 patients who do not have any heart disease.

Frequency distribution of `target` variable wrt `sex`

In [138...
```
df.groupby('sex').target.value_counts()
```

Out[138...
```
sex  target
0    1          72
     0          24
1    0         114
     1          93
Name: count, dtype: int64
```

- `sex` variable contains two integer values 1 and 0 : (1 = male; 0 = female).
- `target` variable also contains two integer values 1 and 0 : (1 = Presence of heart disease; 0 = Absence of heart disease)
- So, out of 96 females - 72 have heart disease and 24 do not have heart disease.
- Similarly, out of 207 males - 93 have heart disease and 114 do not have heart disease.

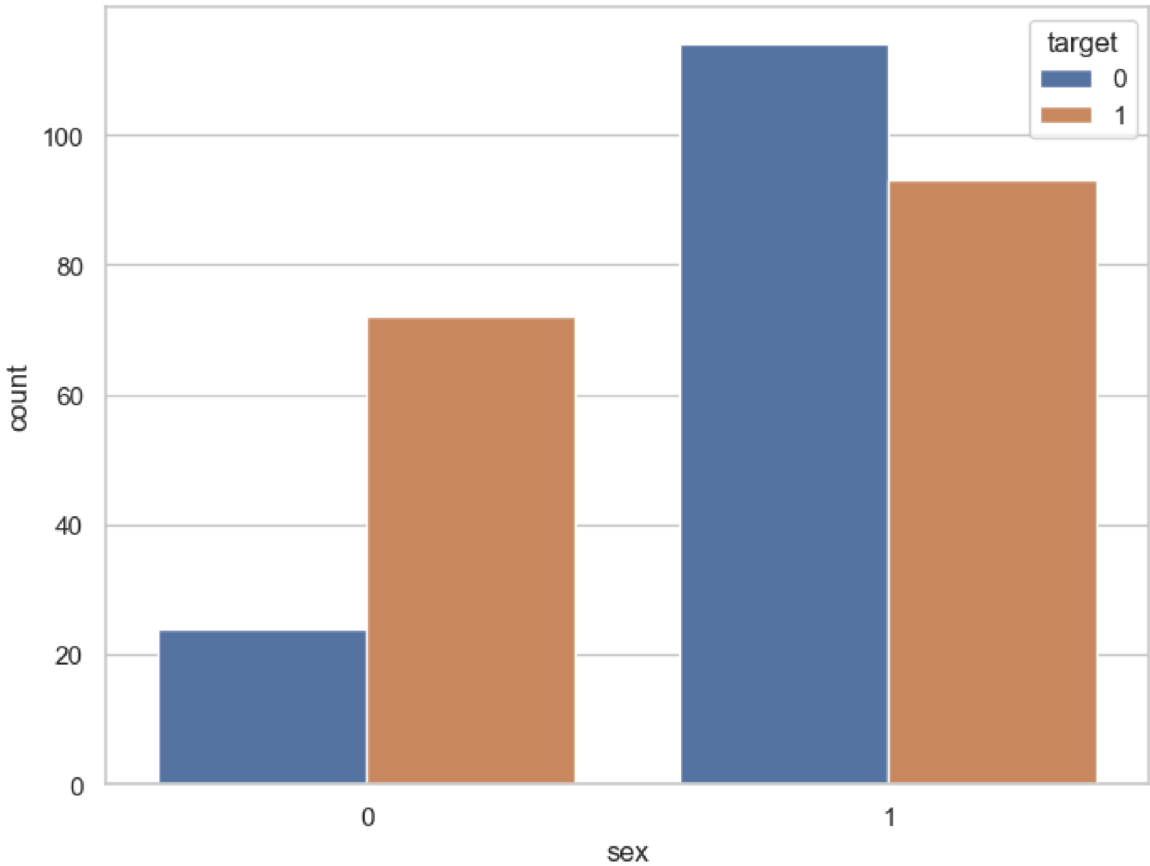
In [140…
```
df.columns
```

Out[140…
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [142…
```
df.groupby('target').sex.value_counts()
```

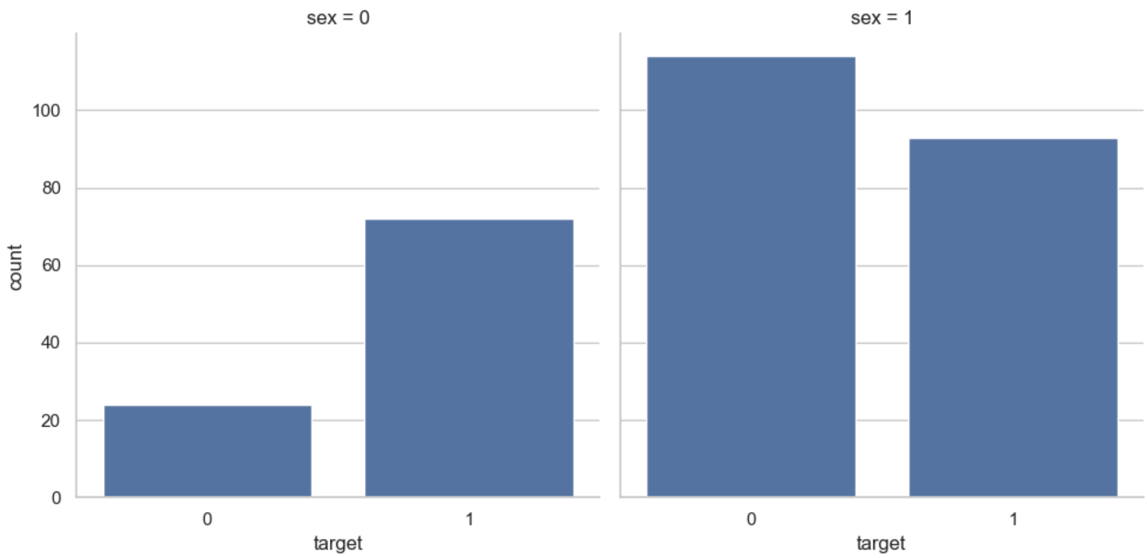
Out[142…
```
target  sex
0       1      114
        0       24
1       1       93
        0       72
Name: count, dtype: int64
```

In [144…
```
ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x='sex',hue='target',data=df)
plt.show()
```

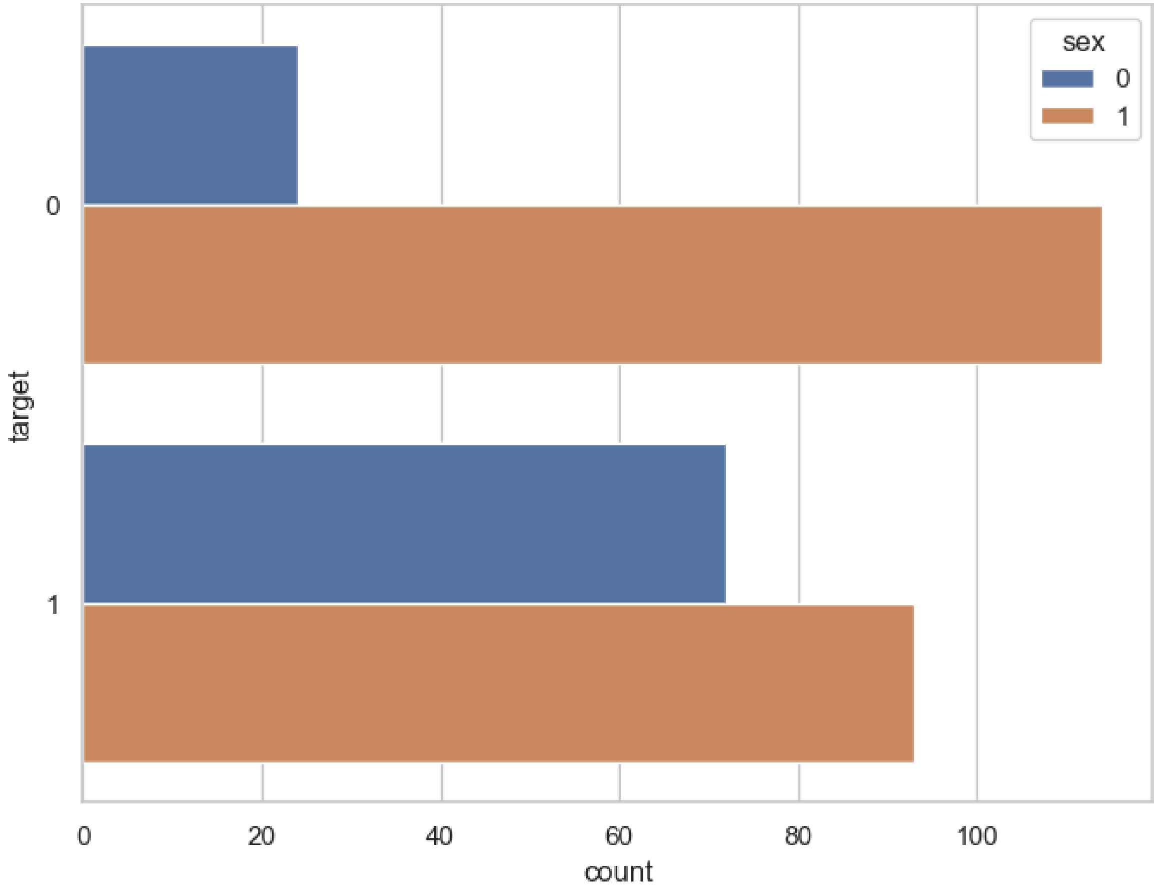


- We can see that the values of `target` variable are plotted wrt `sex` : (1 = male; 0 = female).
- `target` variable also contains two integer values 1 and 0 : (1 = Presence of heart disease; 0 = Absence of heart disease)
- The above plot confirms our findings that -
  - Out of 96 females - 72 have heart disease and 24 do not have heart disease.
  - Similarly, out of 207 males - 93 have heart disease and 114 do not have heart disease.

In [43]:
```
ax= sns.catplot(x="target",col="sex",data=df,kind="count",height=5,aspect=1)
plt.show()
```



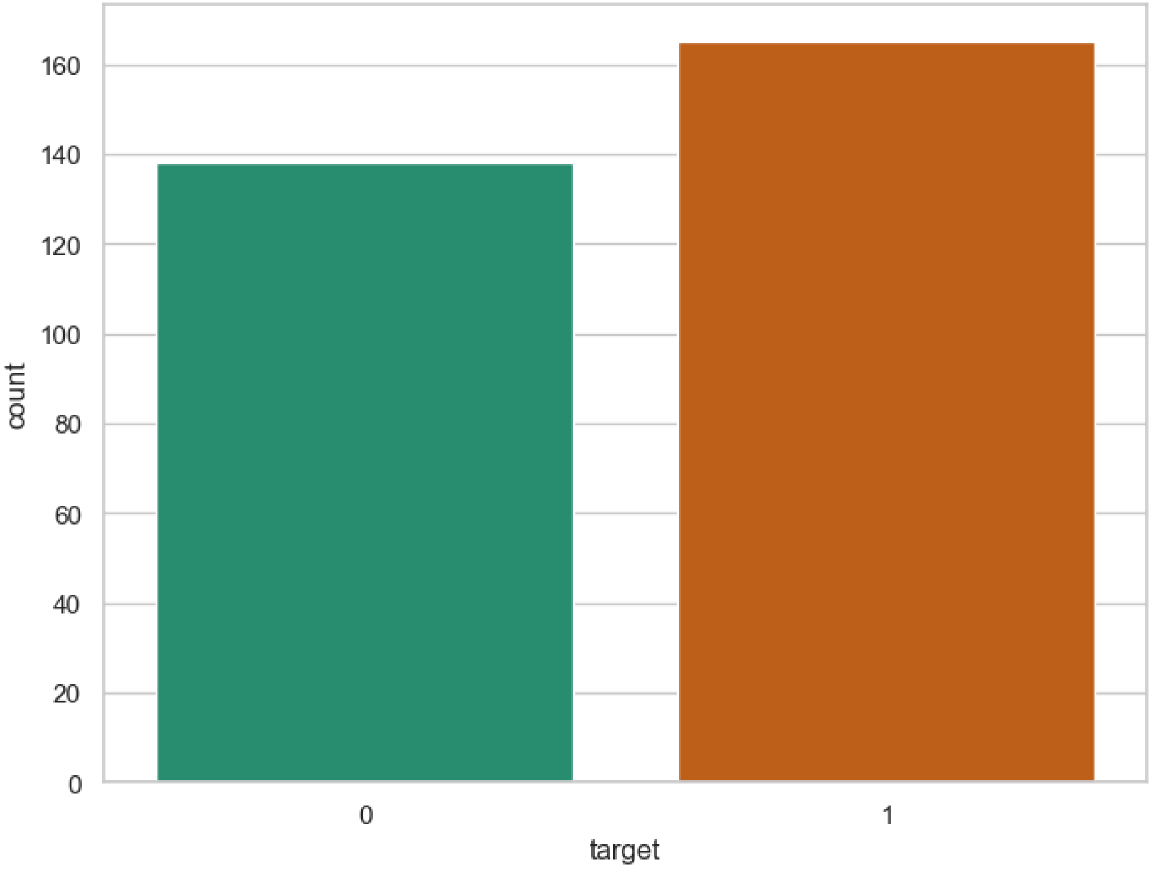
In [146…
```
ax= plt.subplots(figsize=(8,6))
ax= sns.countplot(y="target",hue="sex",data=df)
plt.show()
```



For different color palette

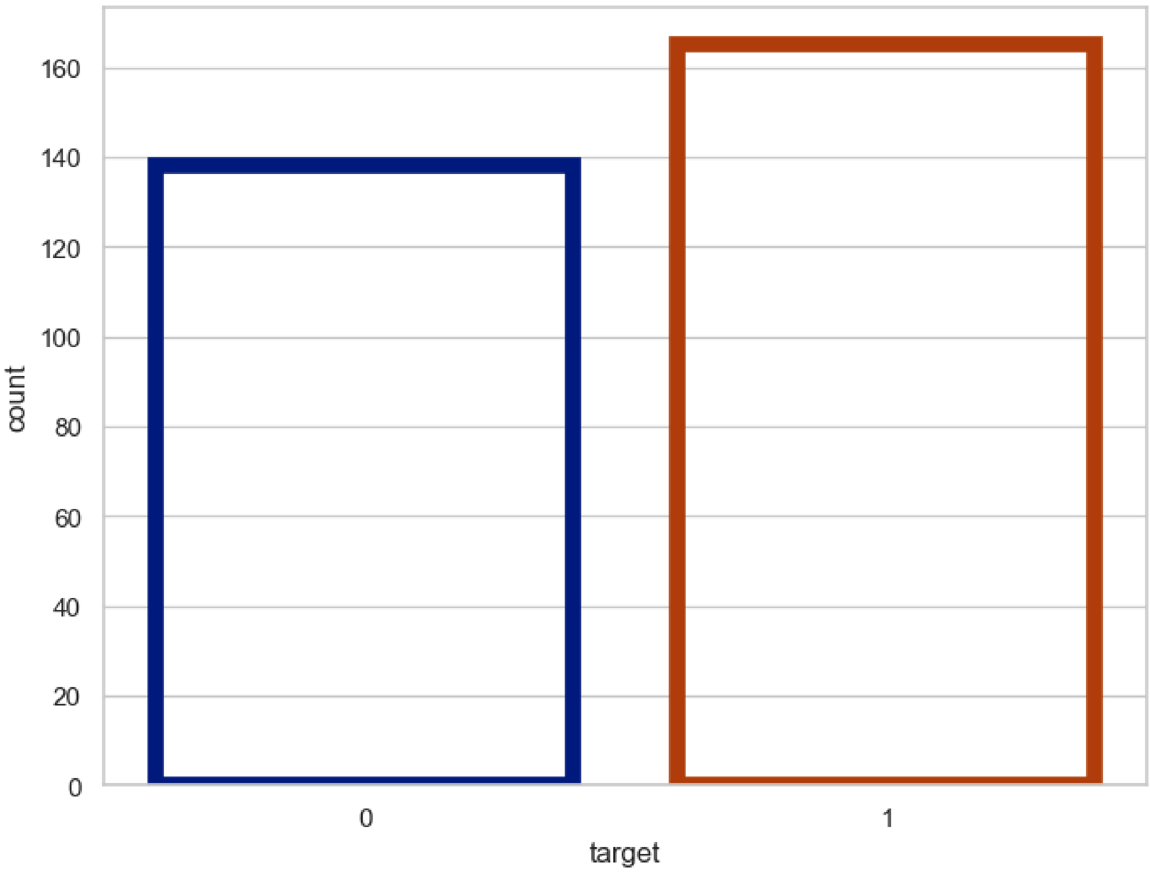
In [148…

```
ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="target",data=df, palette="Dark2")
plt.show()
```



In [150…

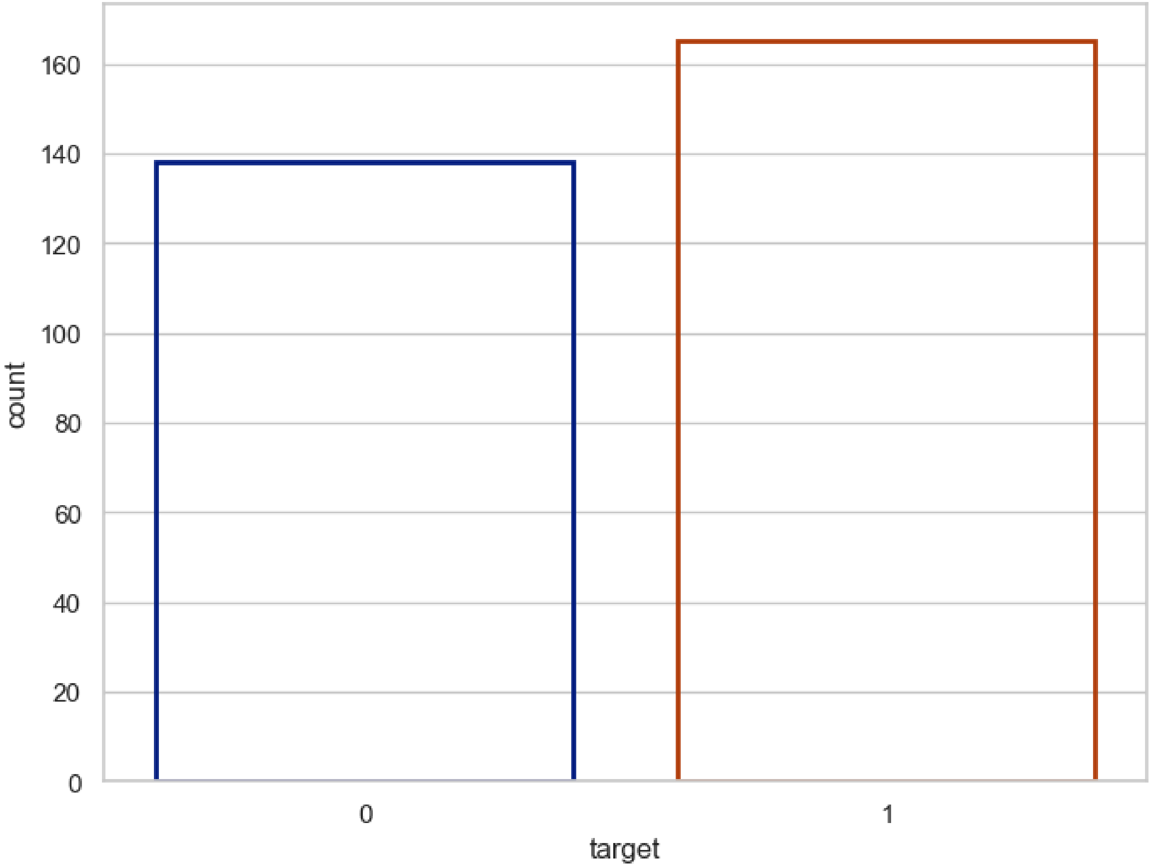
```
ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="target", data=df, facecolor=(0,0,0,0), linewidth=7, edgecolc
plt.show()
```



In [152…

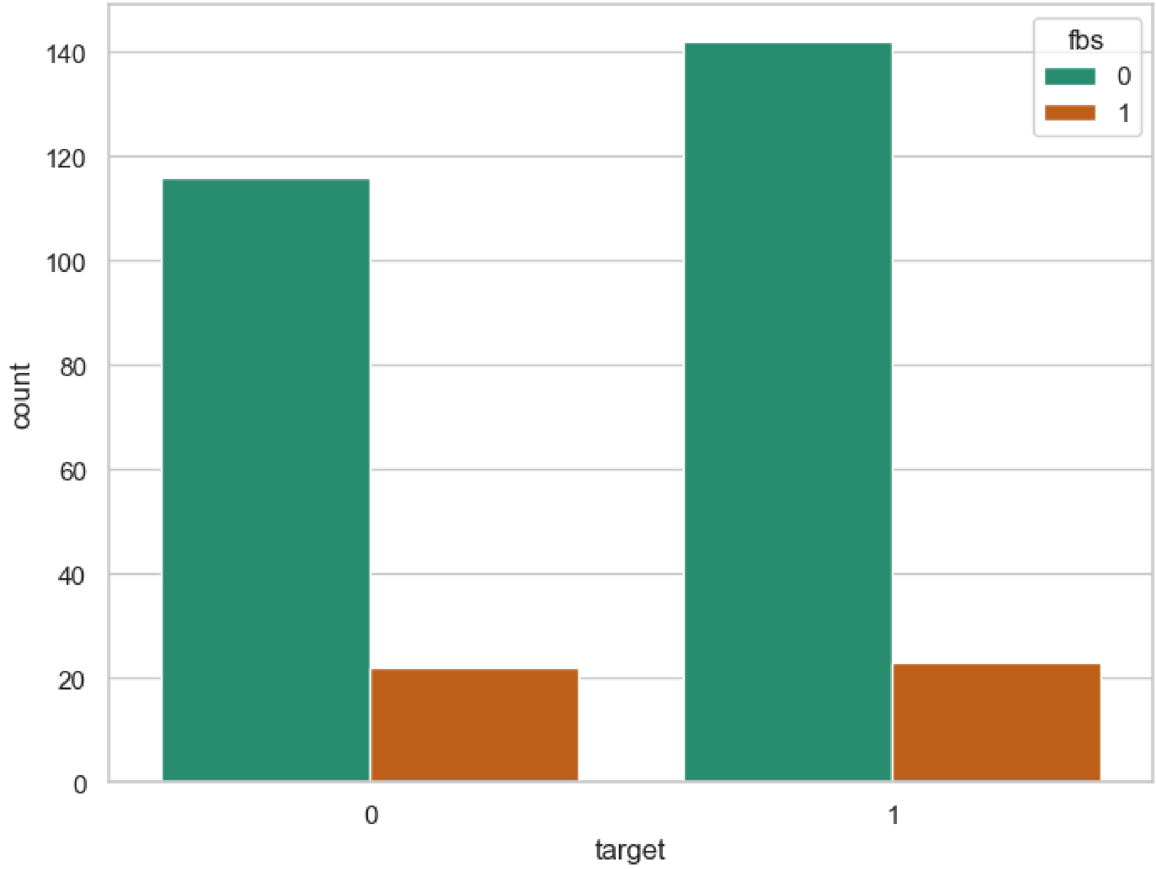
```
ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="target",data=df,facecolor=(0,0,0,0),linewidth=2, edgecolor=s
```

```
plt.show()
```



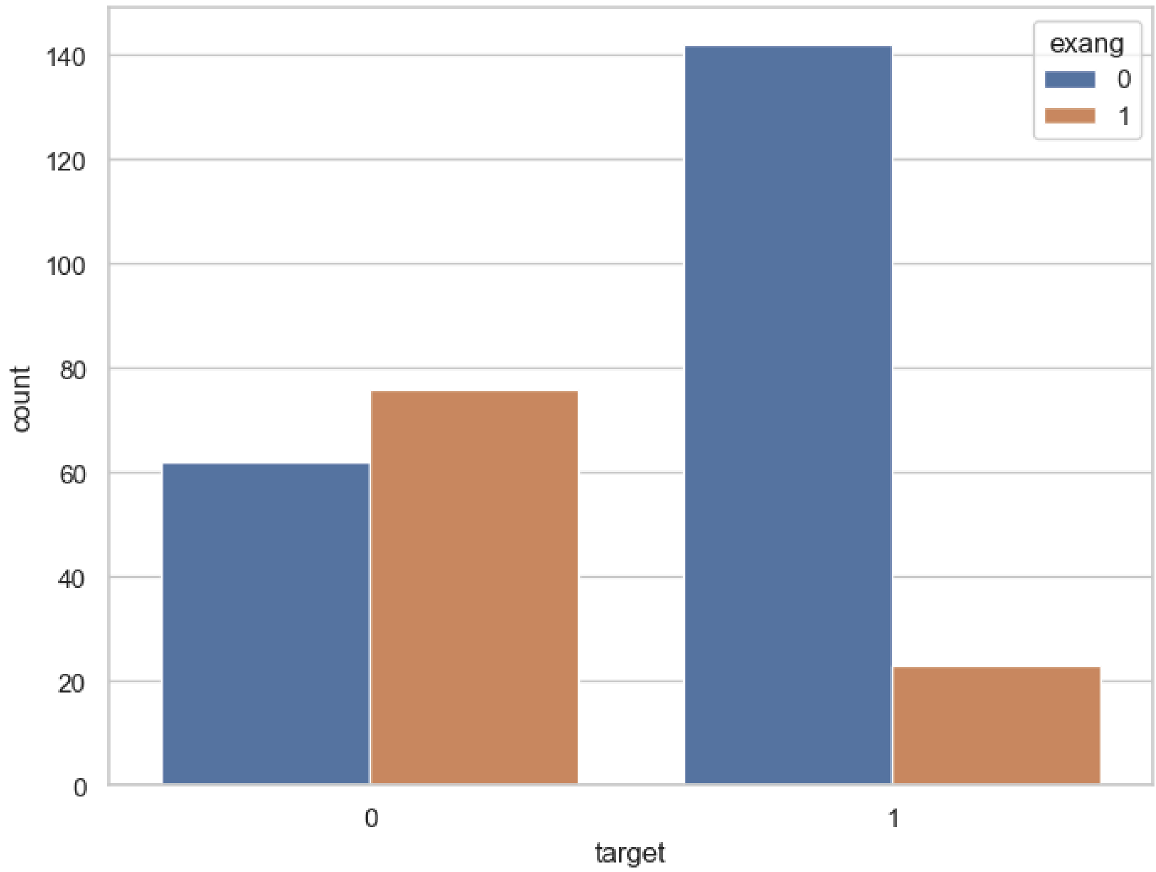
- I have visualize the `target` values distribution wrt `sex`.
- We can follow the same principles and visualize the `target` values distribution wrt `fbs (fasting blood sugar)` and `exang (exercise induced angina)`.

In [154…
```
ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="target",hue="fbs",data=df,palette="Dark2")
plt.show()
```



In [156…

```
ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="target",hue="exang",data=df)
plt.show()
```



- Our feature variable of interest is `target`.

- It refers to the presence of heart disease in the patient.
- It is integer valued as it contains two integers 0 and 1 - (0 stands for absence of heart disease and 1 for presence of heart disease).
- `1` stands for presence of heart disease. So, there are 165 patients suffering from heart disease.
- Similarly, `0` stands for absence of heart disease. So, there are 138 patients who do not have any heart disease.
- There are 165 patients suffering from heart disease, and
- There are 138 patients who do not have any heart disease.
- Out of 96 females - 72 have heart disease and 24 do not have heart disease.
- Similarly, out of 207 males - 93 have heart disease and 114 do not have heart disease.

In [158…
```
df.columns
```

Out[158…
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

Estimate correlation coefficients

Our dataset is very small. So, I will compute the standard correlation coefficient (also called Pearson's r) between every pair of attributes. I will compute it using the `df.corr()` method as follows:-

It's a quick way to explore patterns and relationships in your data during exploratory data analysis (EDA).

In [160…
```
correlation=df.corr()
```

In [162…
```
correlation.target.sort_values(ascending=False)
```

Out[162…

```
target      1.000000
cp          0.433798
thalach     0.421741
slope       0.345877
restecg     0.137230
fbs        -0.028046
chol       -0.085239
trestbps   -0.144931
age        -0.225439
sex        -0.280937
thal       -0.344029
ca         -0.391724
oldpeak    -0.430696
exang      -0.436757
Name: target, dtype: float64
```

Interpretation of correlation coefficient

- The correlation coefficient ranges from -1 to +1.
- When it is close to +1, this signifies that there is a strong positive correlation. So, we can see that there is no variable which has strong positive correlation with `target` variable.
- When it is clsoe to -1, it means that there is a strong negative correlation. So, we can see that there is no variable which has strong negative correlation with `target` variable.
- When it is close to 0, it means that there is no correlation. So, there is no correlation between `target` and `fbs` .
- We can see that the `cp` and `thalach` variables are mildly positively correlated with `target` variable. So, I will analyze the interaction between these features and `target` variable.

In [164…

```
df.cp.nunique()
```

Out[164…

```
4
```

- It can be seen that `cp` is a categorical variable and it contains 4 types of values - 0, 1, 2 and 3.

In [166…

```
df.cp.unique()
```

Out[166…

```
array([3, 2, 1, 0], dtype=int64)
```

In [168…

```
df.cp.value_counts()
```

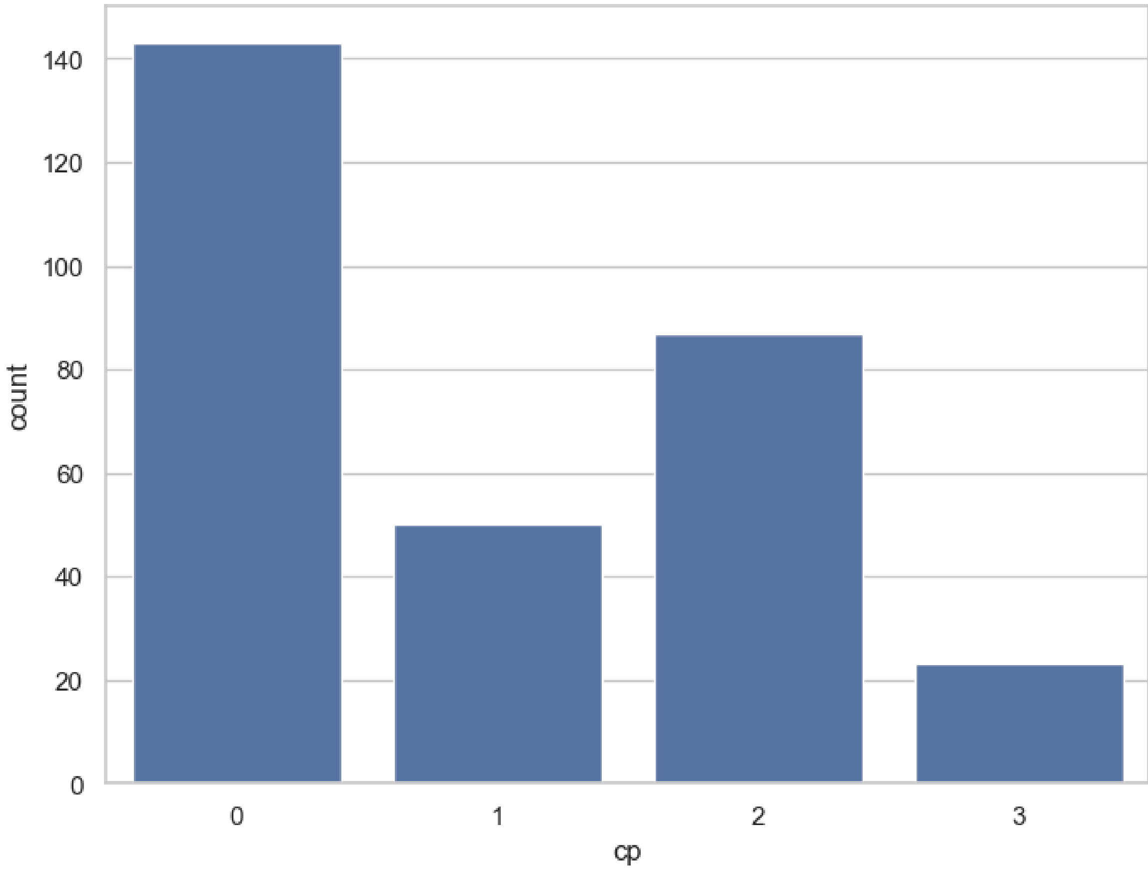
Out[168…

```
cp
0    143
2     87
1     50
3     23
Name: count, dtype: int64
```

- `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.

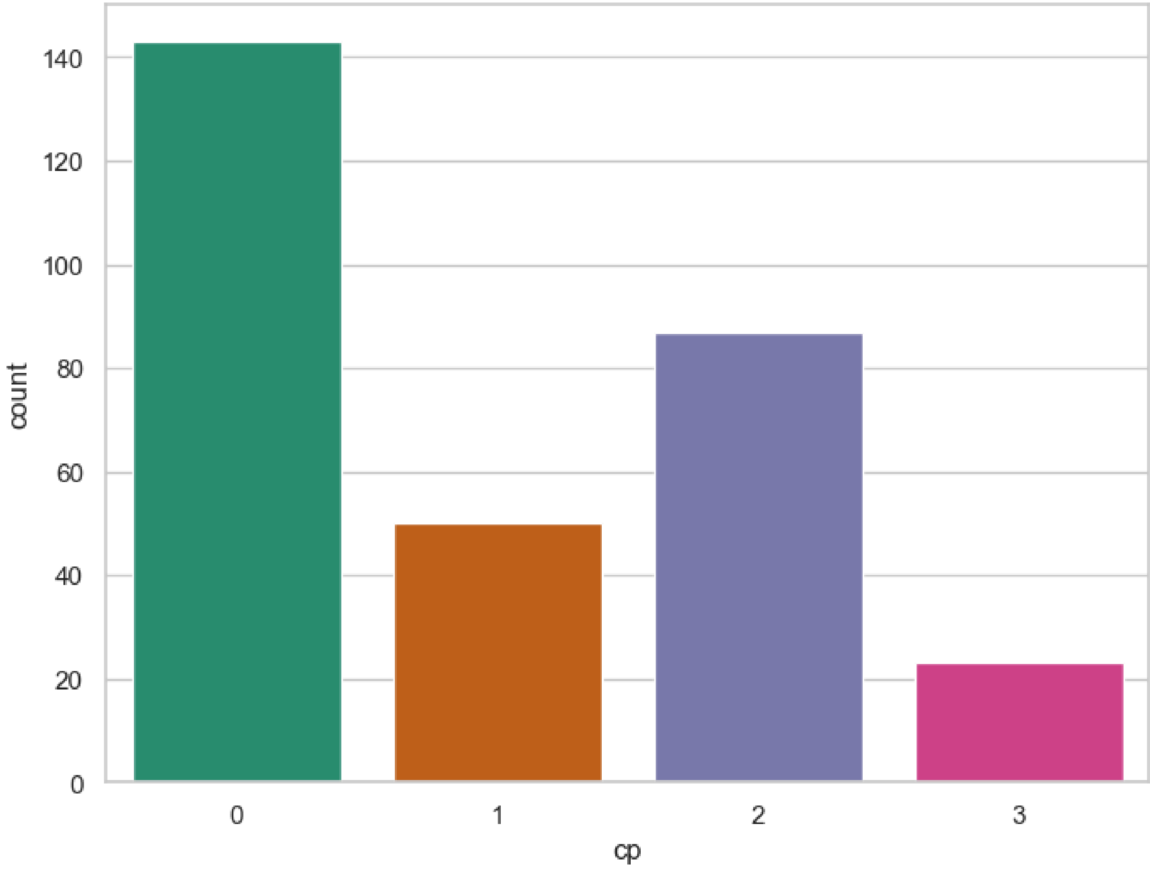
In [176…

```
ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="cp",data=df)
plt.show()
```



In [178…

```
ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="cp",data=df,palette="Dark2")
plt.show()
```



In [174…

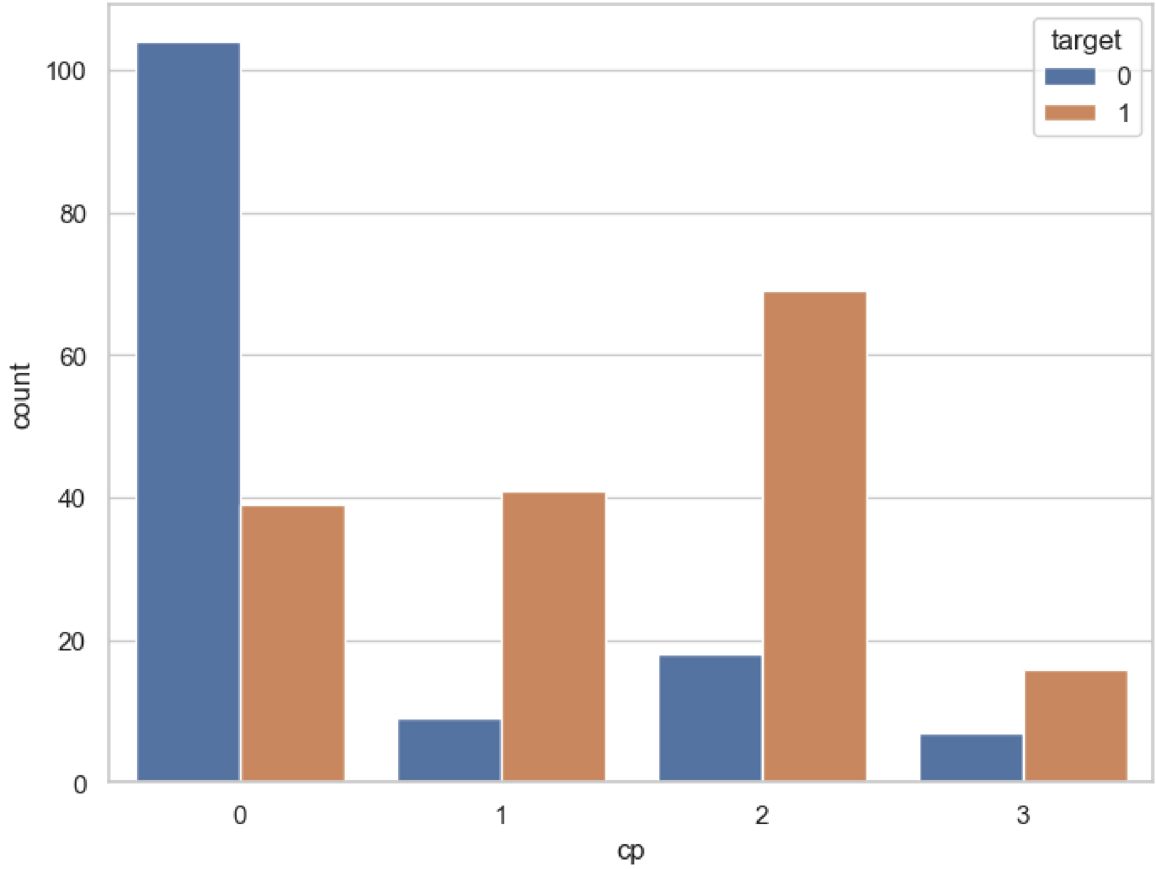
```
df.groupby('cp').target.value_counts()
```

Out[174…

```
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
```

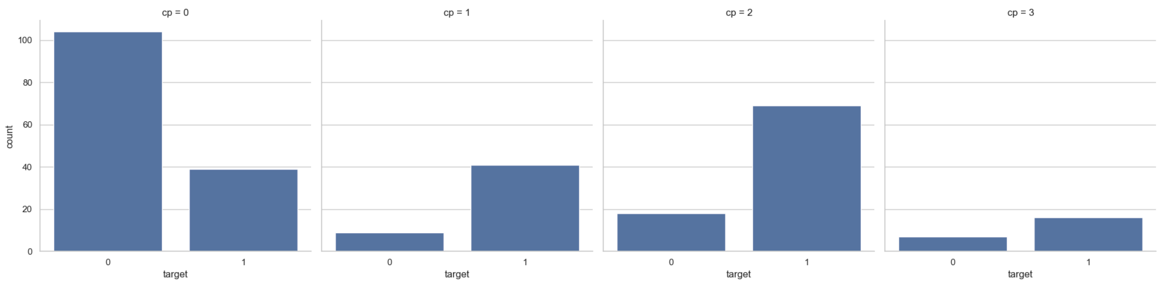
In [180…

```
ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="cp",hue="target",data=df)
plt.show()
```

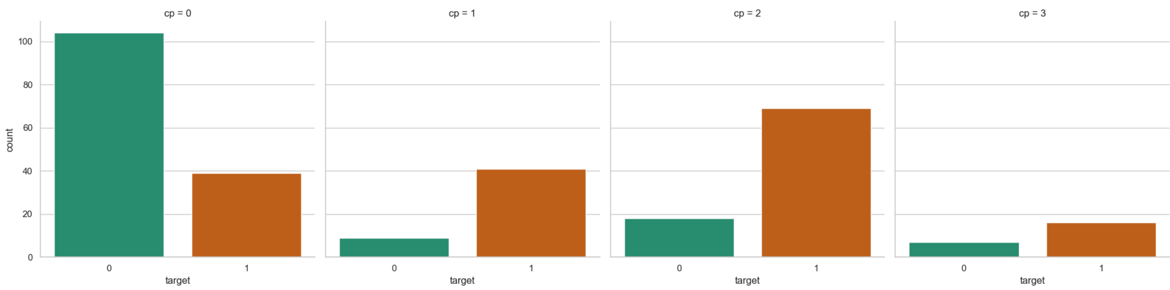


- We can see that the values of `target` variable are plotted wrt `cp`.
- `target` variable contains two integer values 1 and 0 : (1 = Presence of heart disease; 0 = Absence of heart disease)
- The above plot confirms our above findings,

In [77]:
```
ax=sns.catplot(x="target",col="cp",data=df,kind="count",height=5,aspect=1)
plt.show()
```



In [81]:
```
ax=sns.catplot(x="target",col="cp",data=df,kind="count",height=5,aspect=1,palett
plt.show()
```



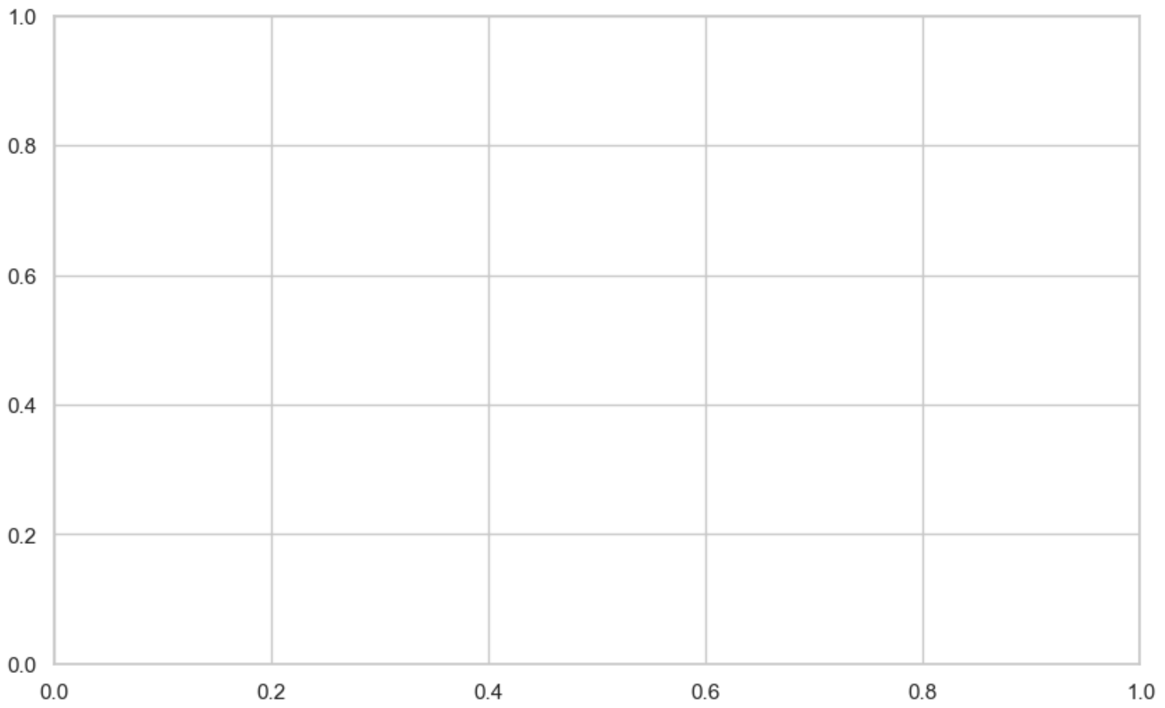
In [82]:
```
df.thalach.nunique()
```

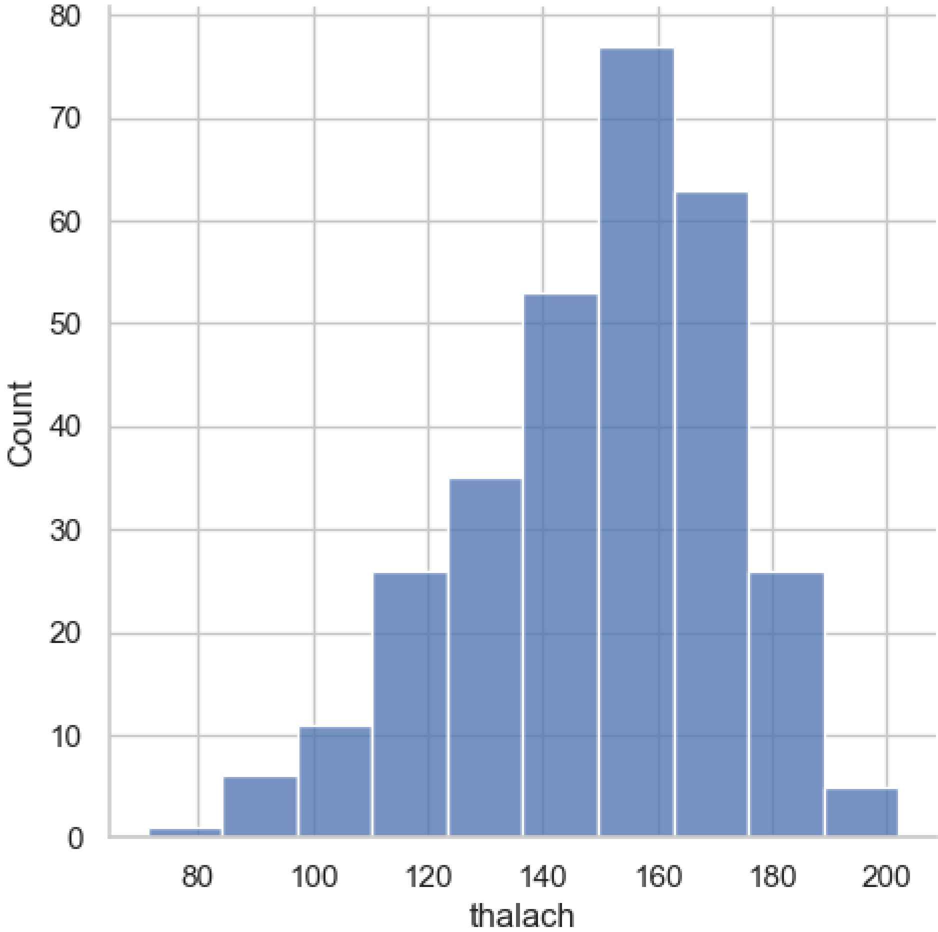
Out[82]:
```
91
```

In [83]:
```
df.thalach.unique()
```

Out[83]:
```
array([150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 171,
       144, 158, 114, 151, 161, 179, 137, 157, 123, 152, 168, 140, 188,
       125, 170, 165, 142, 180, 143, 182, 156, 115, 149, 146, 175, 186,
       185, 159, 130, 190, 132, 147, 154, 202, 166, 164, 184, 122, 169,
       138, 111, 145, 194, 131, 133, 155, 167, 192, 121,  96, 126, 105,
       181, 116, 108, 129, 120, 112, 128, 109, 113,  99, 177, 141, 136,
        97, 127, 103, 124,  88, 195, 106,  95, 117,  71, 118, 134,  90],
      dtype=int64)
```

In [182…
```
ax=plt.subplots(figsize=(10,6))
ax=sns.displot(x="thalach",data=df,bins=10)
plt.show()
```

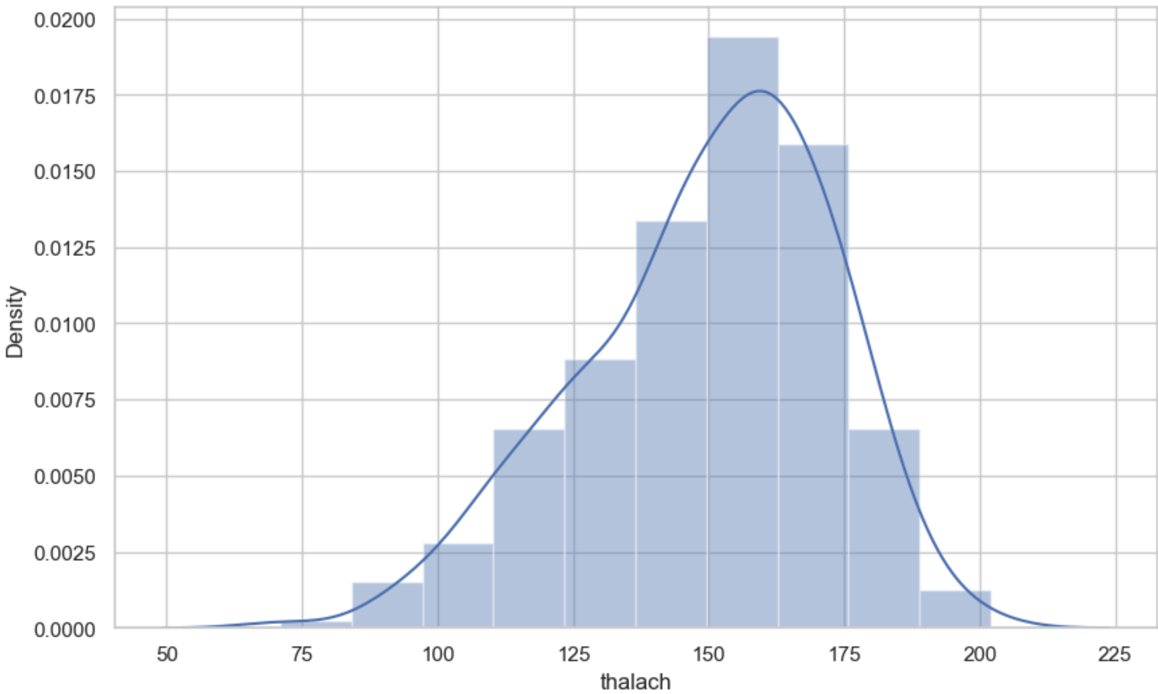




Visualize the frequency distribution of `thalach` variable

In [184…

```
ax=plt.subplots(figsize=(10,6))
x=df.thalach
ax=sns.distplot(x,bins=10)
plt.show()
```

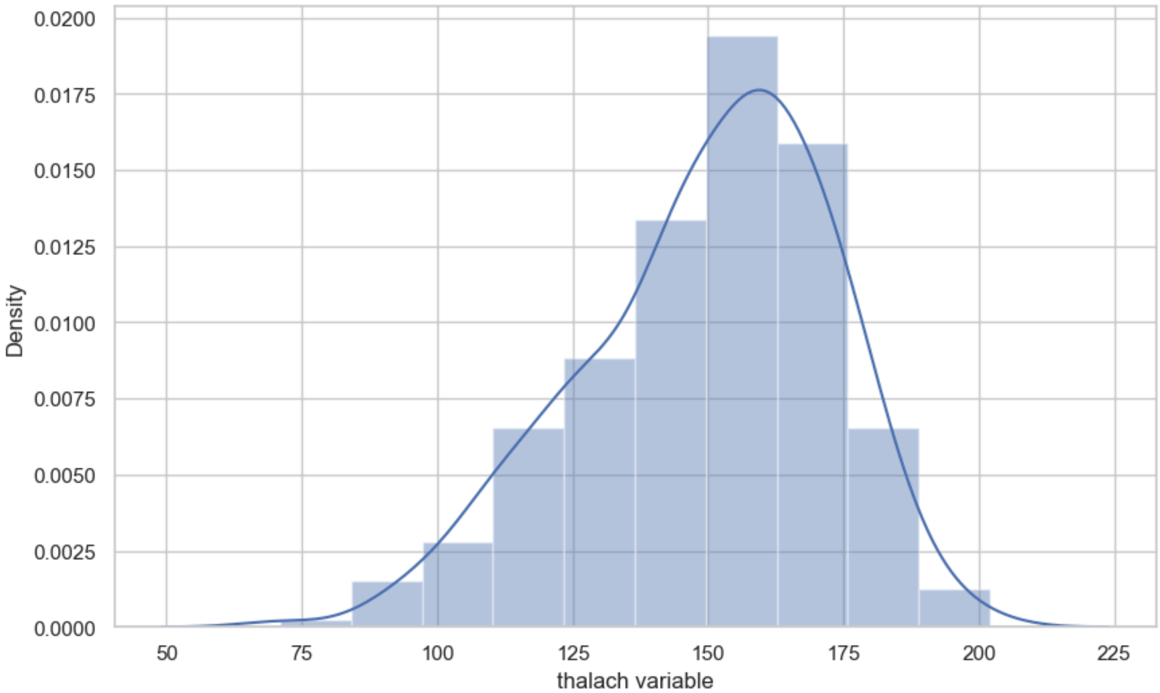


- We can see that the `thalach` variable is slightly negatively skewed.

We can use Pandas series object to get an informative axis label as follows :

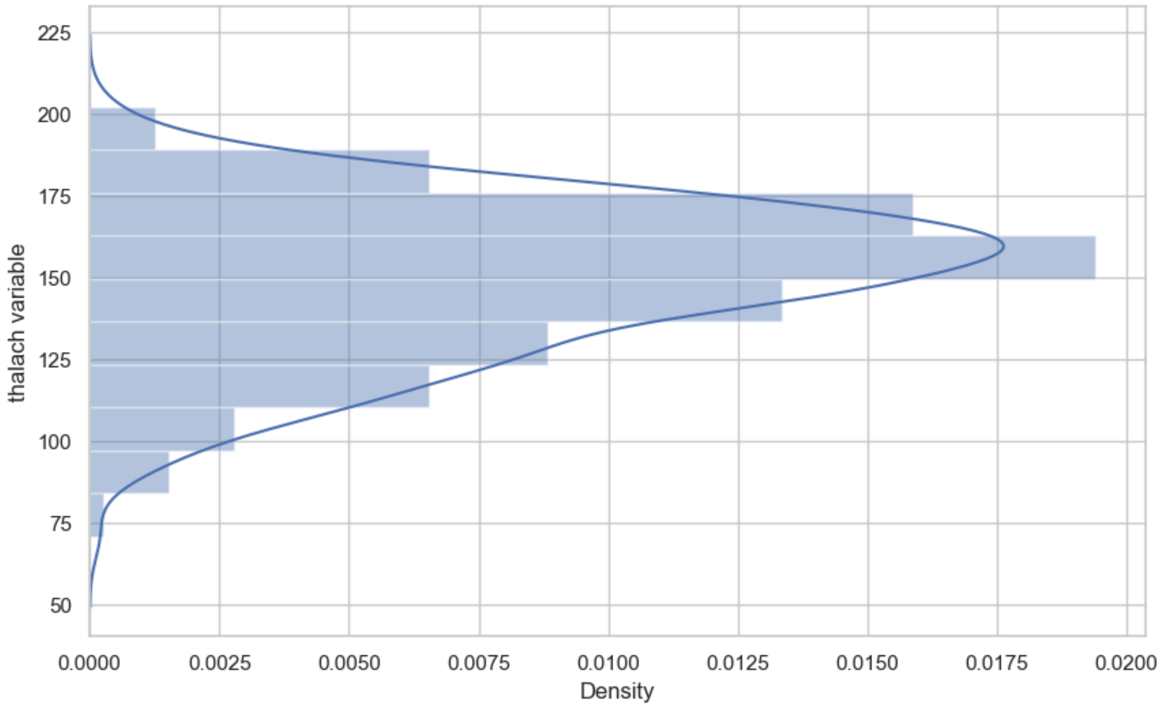
In [186…

```
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 [188…

```
ax=plt.subplots(figsize=(10,6))
x=df.thalach
x=pd.Series(x, name="thalach variable")
ax = sns.distplot(x,bins=10,vertical=True)
plt.show()
```

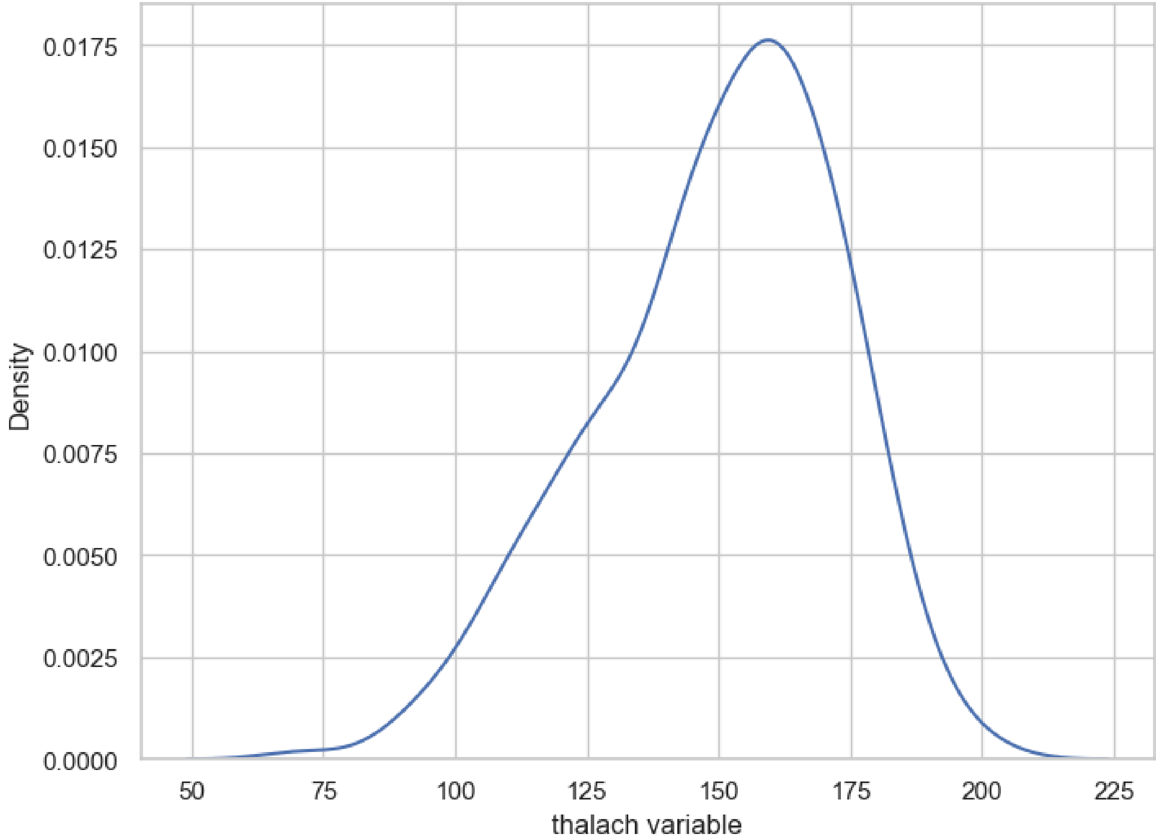


Seaborn Kernel Density Estimation (KDE) Plot

- The kernel density estimate (KDE) plot is a useful tool for plotting the shape of a distribution.
- The KDE plot plots the density of observations on one axis with height along the other axis.
- We can plot a KDE plot as follows :

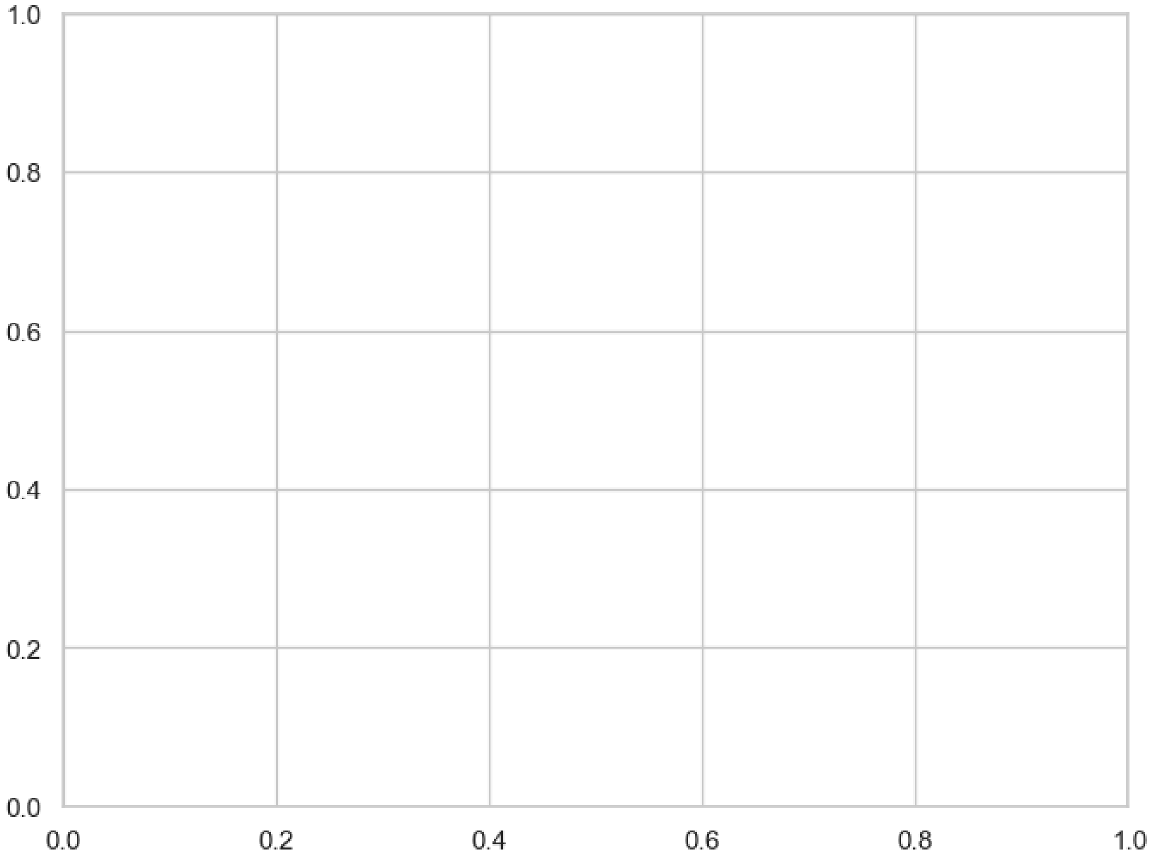
In [190…

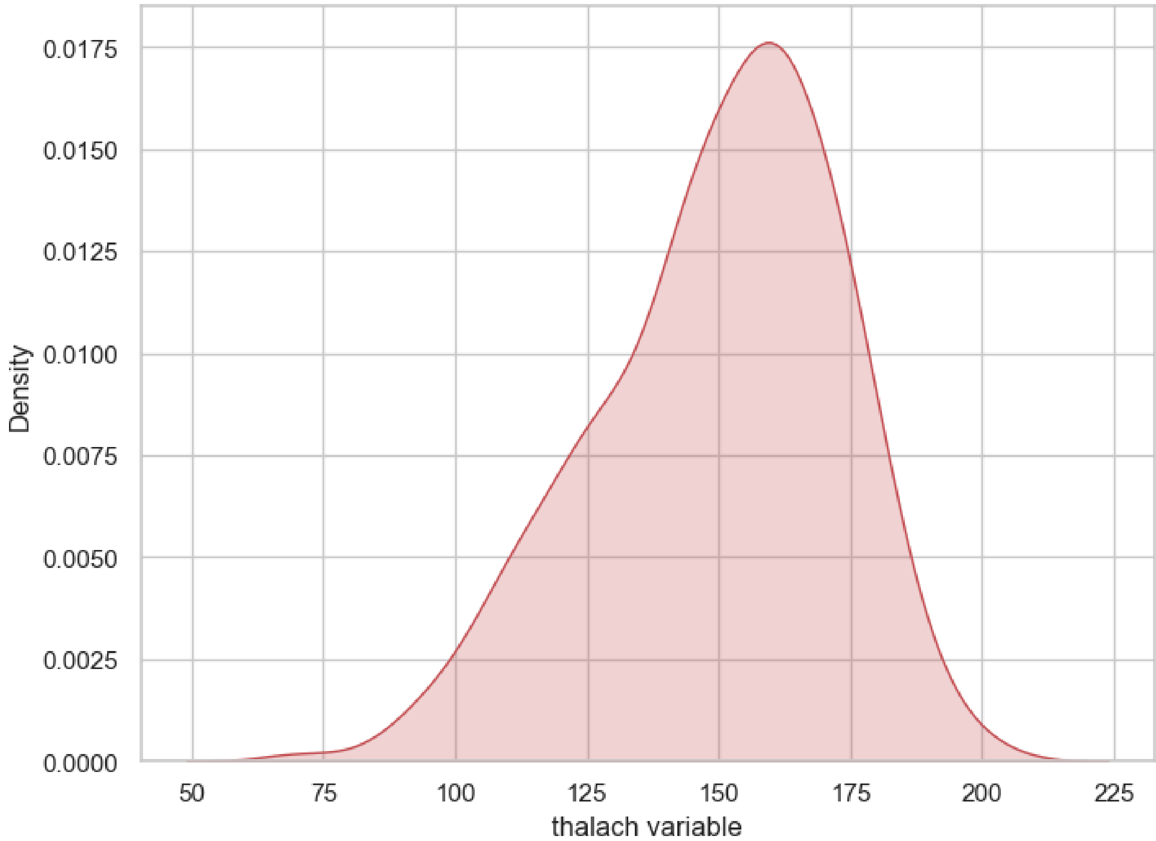
```
ax=plt.subplots(figsize=(8,6))
x=df.thalach
x=pd.Series(x, name="thalach variable")
ax=sns.kdeplot(x)
plt.show()
```



In [200…

```
ax=plt.subplots(figsize=(8,6))
x=df.thalach
x=pd.Series(x, name="thalach variable")
ax=sns.kdeplot(x,shade=True, color='r')
plt.show()
```



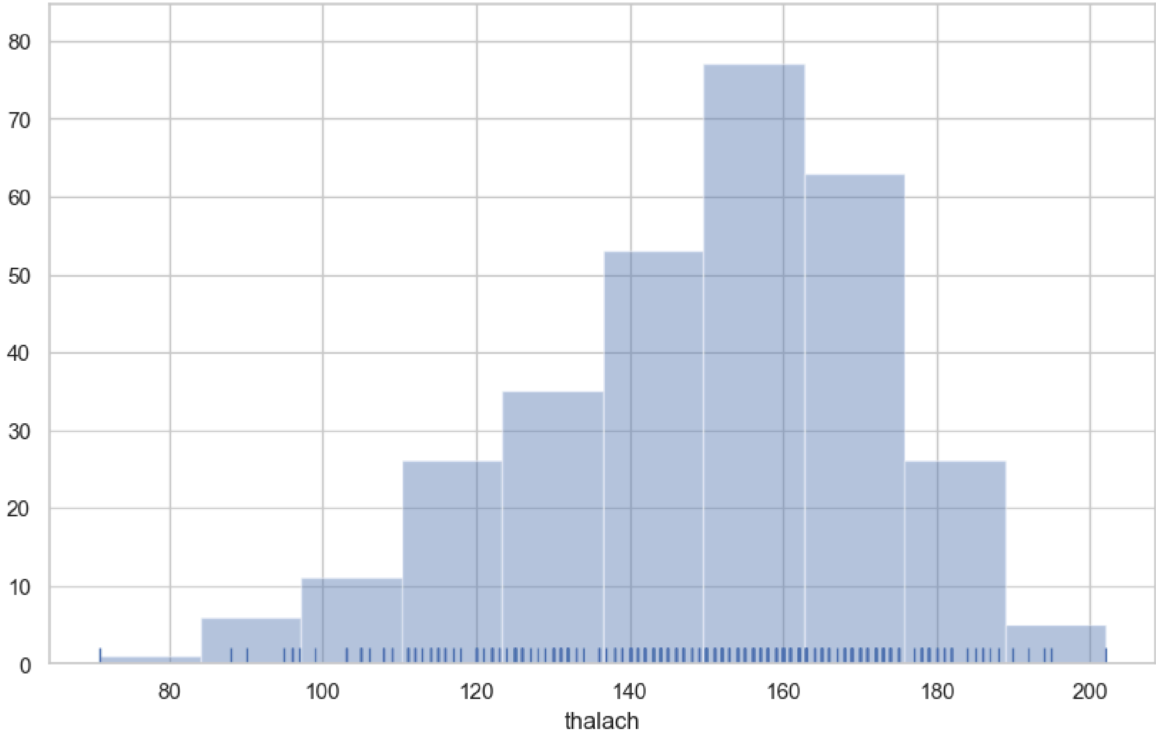


Histogram

- A histogram represents the distribution of data by forming bins along the range of the data and then drawing bars to show the number of observations that fall in each bin.
- We can plot a histogram as follows :

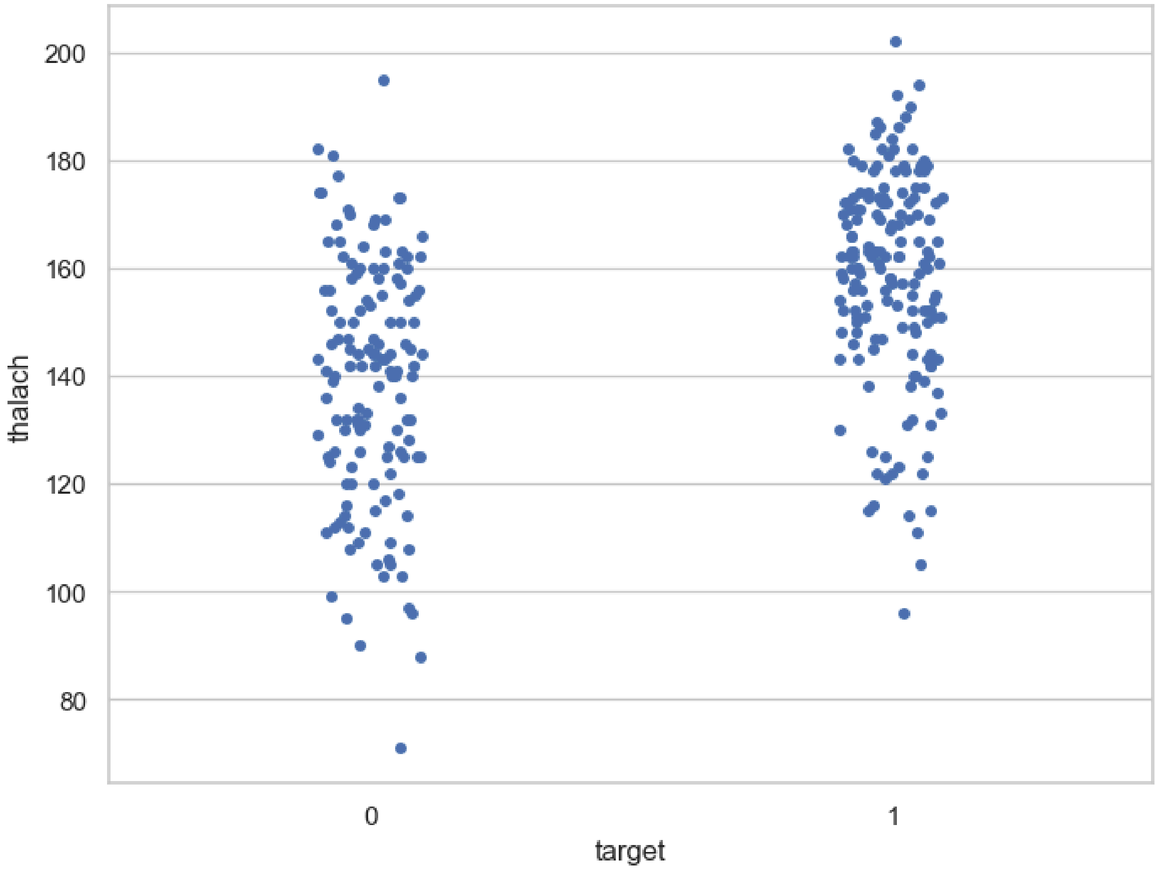
In [216…

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



In [218…

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

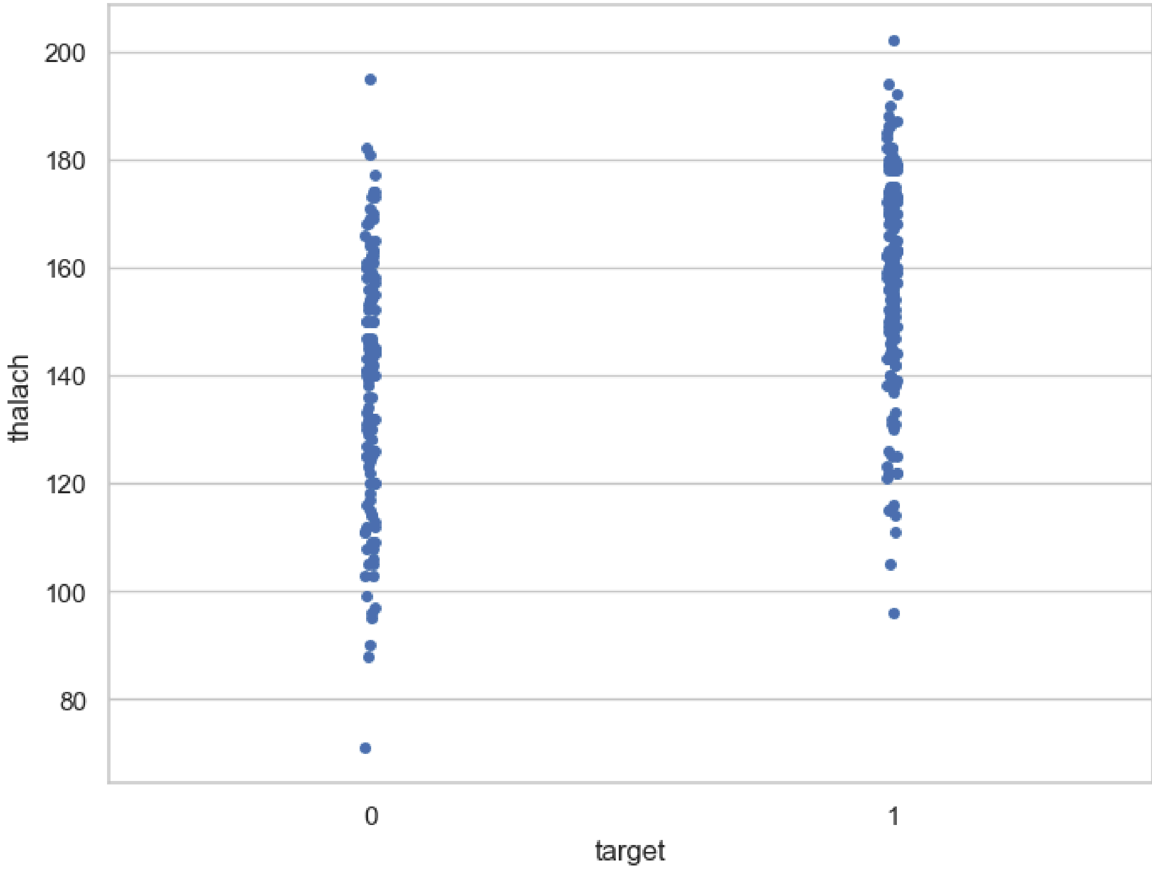


Interpretation

- 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).

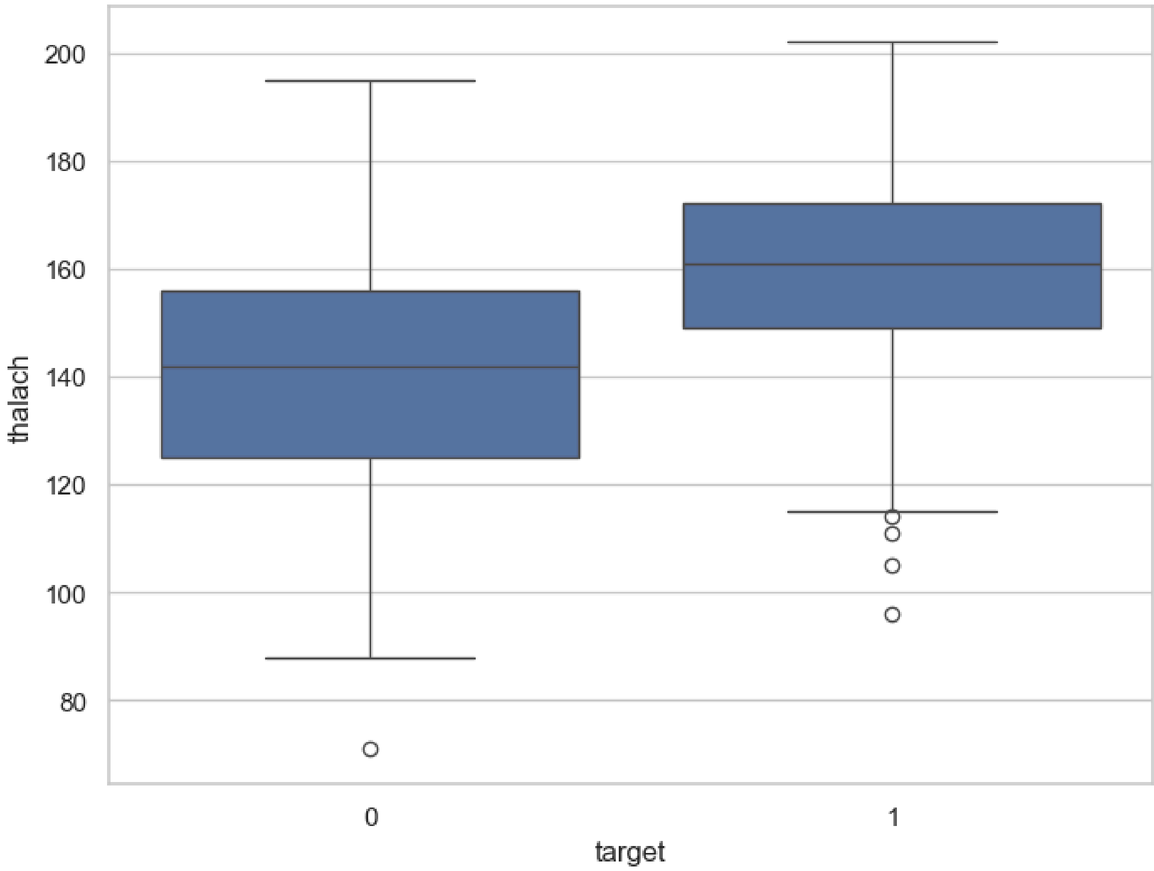
In [228…

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



In [230…

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



- 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).

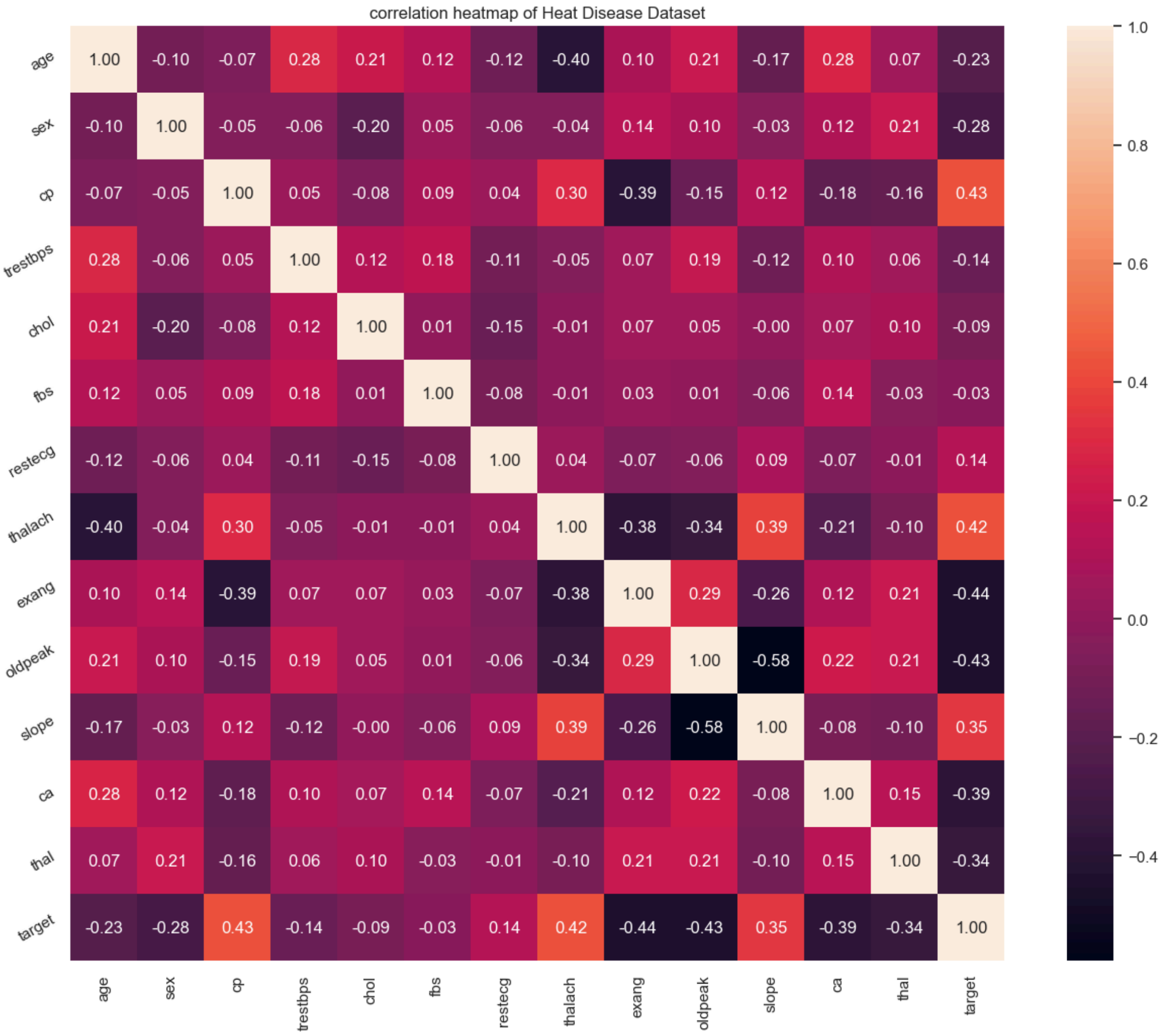
Discover patterns and relationships

- An important step in EDA is to discover patterns and relationships between variables in the dataset.
- I will use `heat map` and `pair plot` to discover the patterns and relationships in the dataset.

Heat Map

In [238…

```
plt.figure(figsize=(16,12))
plt.title('correlation heatmap of Heat Disease Dataset')
a=sns.heatmap(correlation, square=True, annot=True, fmt='.2f', linecolor='white'
a.set_xticklabels(a.get_xticklabels(), rotation=90)
a.set_yticklabels(a.get_yticklabels(), rotation=30)
plt.show()
```



Interpretation

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

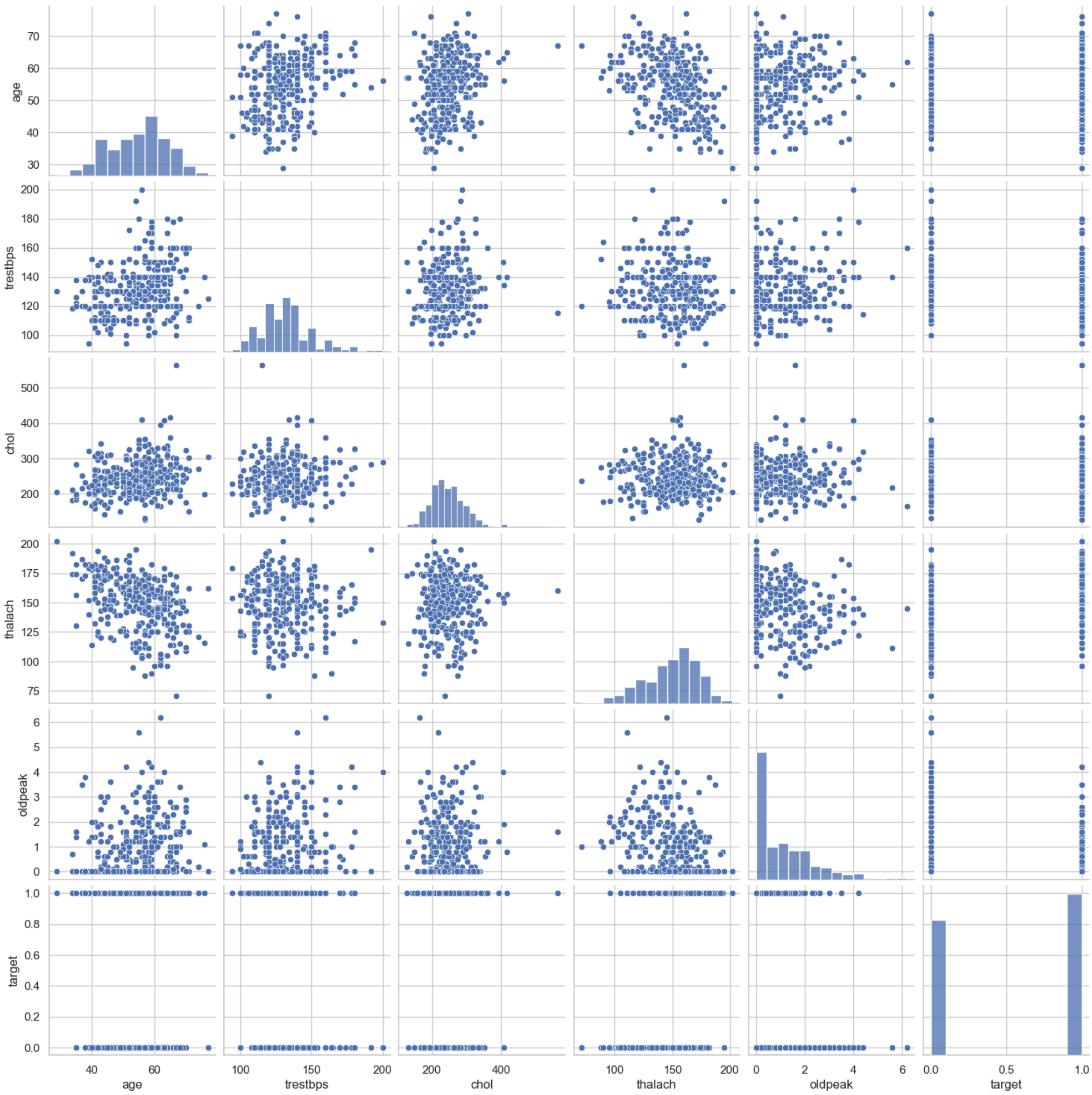
- `target` and `cp` variable are mildly positively correlated (correlation coefficient = 0.43).

- `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

In [244…

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



- 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 variables

In [246…
```
df.age.nunique()
```

Out[246…
```
41
```

In [248…
```
df.age.unique()
```

Out[248…
```
array([63, 37, 41, 56, 57, 44, 52, 54, 48, 49, 64, 58, 50, 66, 43, 69, 59,
       42, 61, 40, 71, 51, 65, 53, 46, 45, 39, 47, 62, 34, 35, 29, 55, 60,
       67, 68, 74, 76, 70, 38, 77], dtype=int64)
```

In [250…
```
df.age.describe()
```

Out[250…

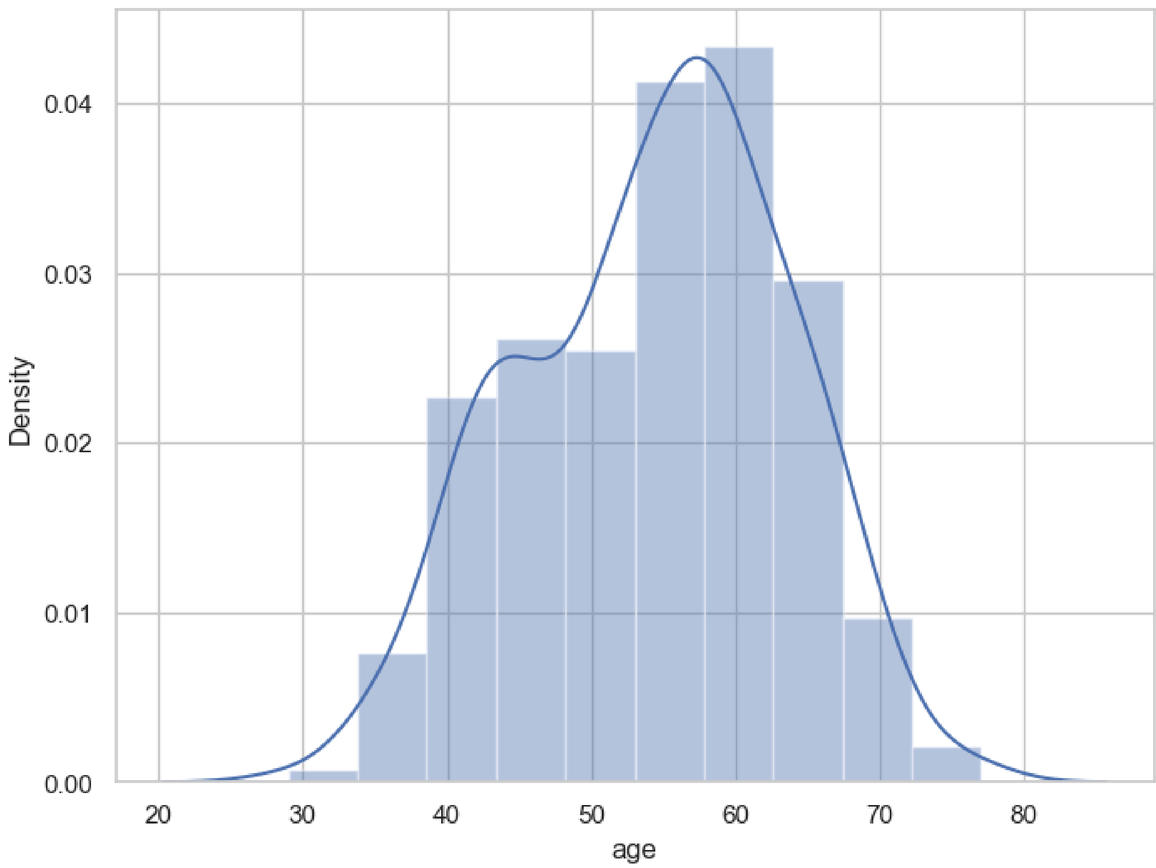
```
count    303.000000
mean      54.366337
std        9.082101
min       29.000000
25%       47.500000
50%       55.000000
75%       61.000000
max       77.000000
Name: age, dtype: float64
```

- The mean value of the `age` variable is 54.37 years.
- The minimum and maximum values of `age` are 29 and 77 years.

Plot the distribution of `age` variable

In [256…

```
ax=plt.subplots(figsize=(8,6))
x=df.age
ax=sns.distplot(x, bins=10)
plt.show()
```



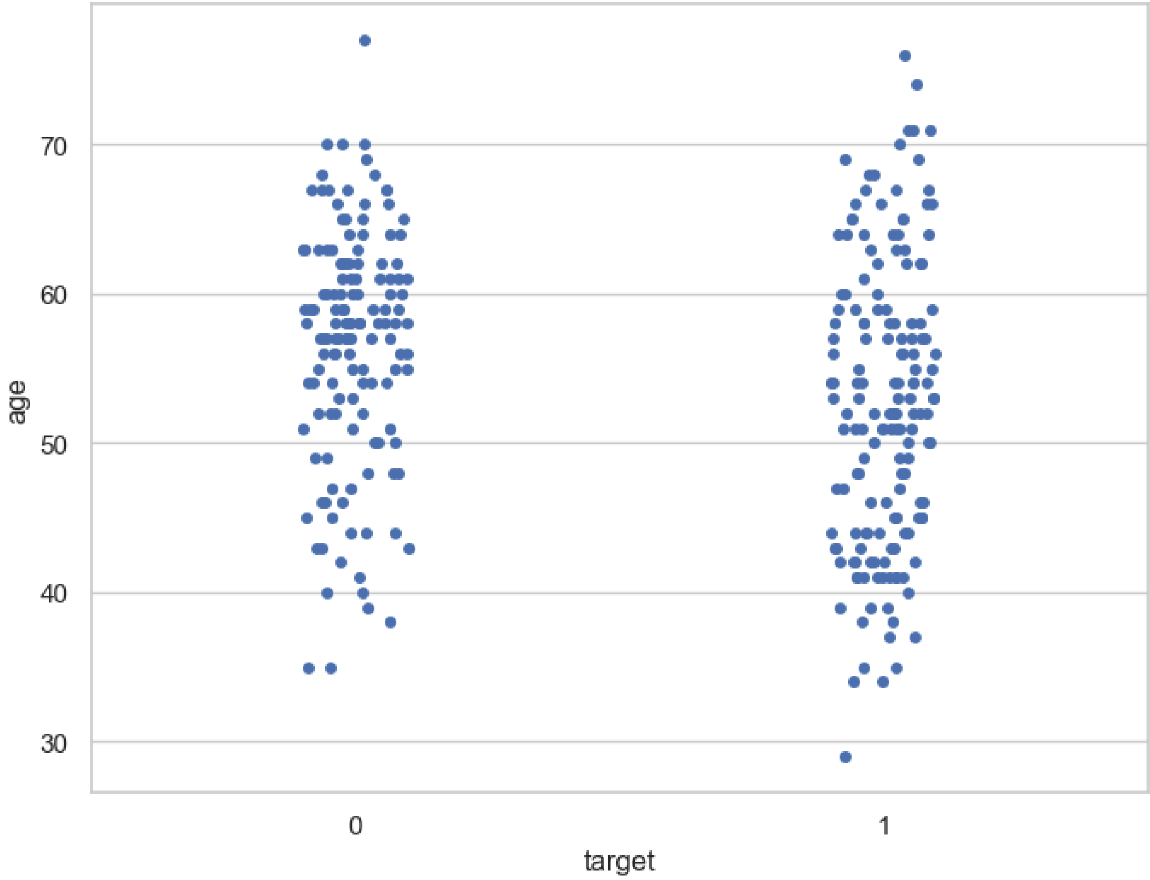
- The `age` variable distribution is approximately normal.

Analyze `age` and `target` variable

In [264…

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

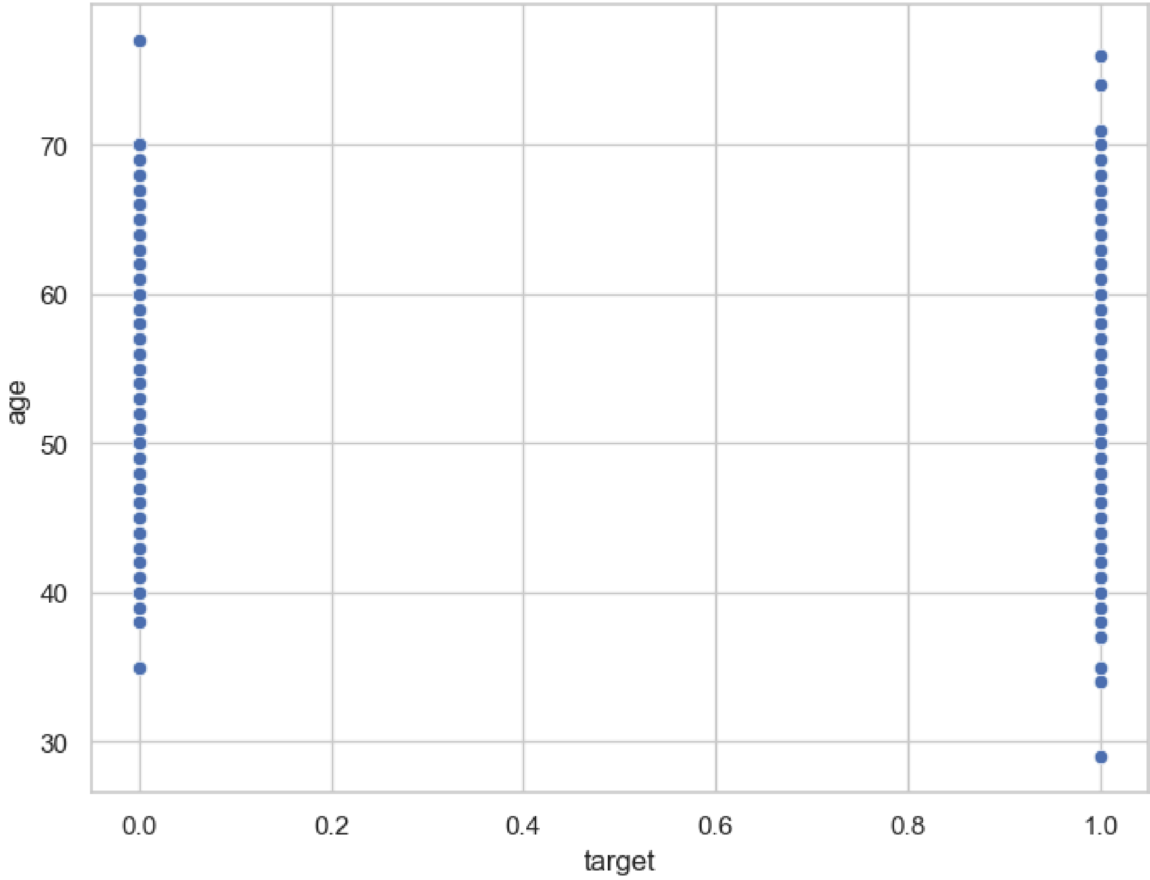
```
plt.show()
```



- We can see that the people suffering from heart disease (target = 1) and people who are not suffering from heart disease (target = 0) have comparable ages.

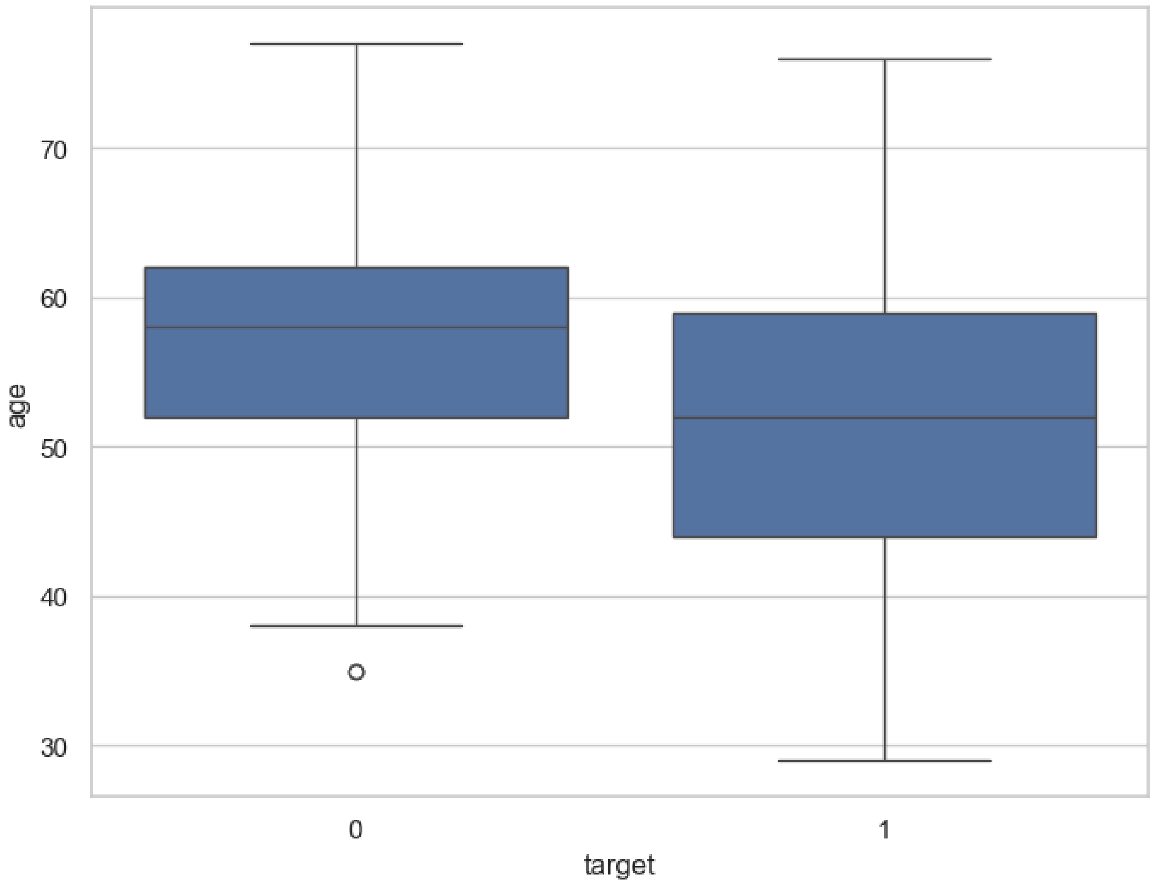
In [278…

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



In [268…

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



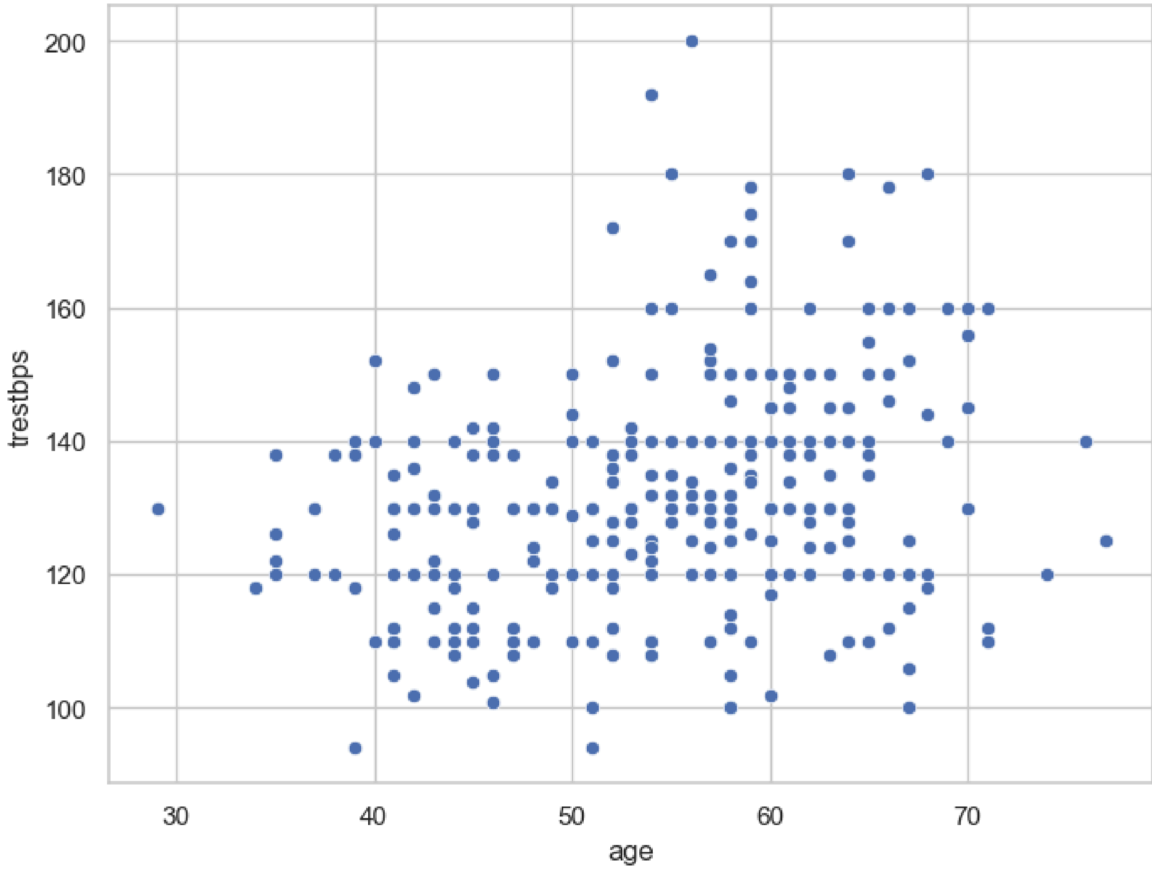
The above boxplot tells two different things :

- The mean age of the people who have heart disease is less than the mean age of the people who do not have heart disease.
- The dispersion or spread of age of the people who have heart disease is greater than the dispersion or spread of age of the people who do not have heart disease.

Analyze `age` and `trestbps` variable

In [274…

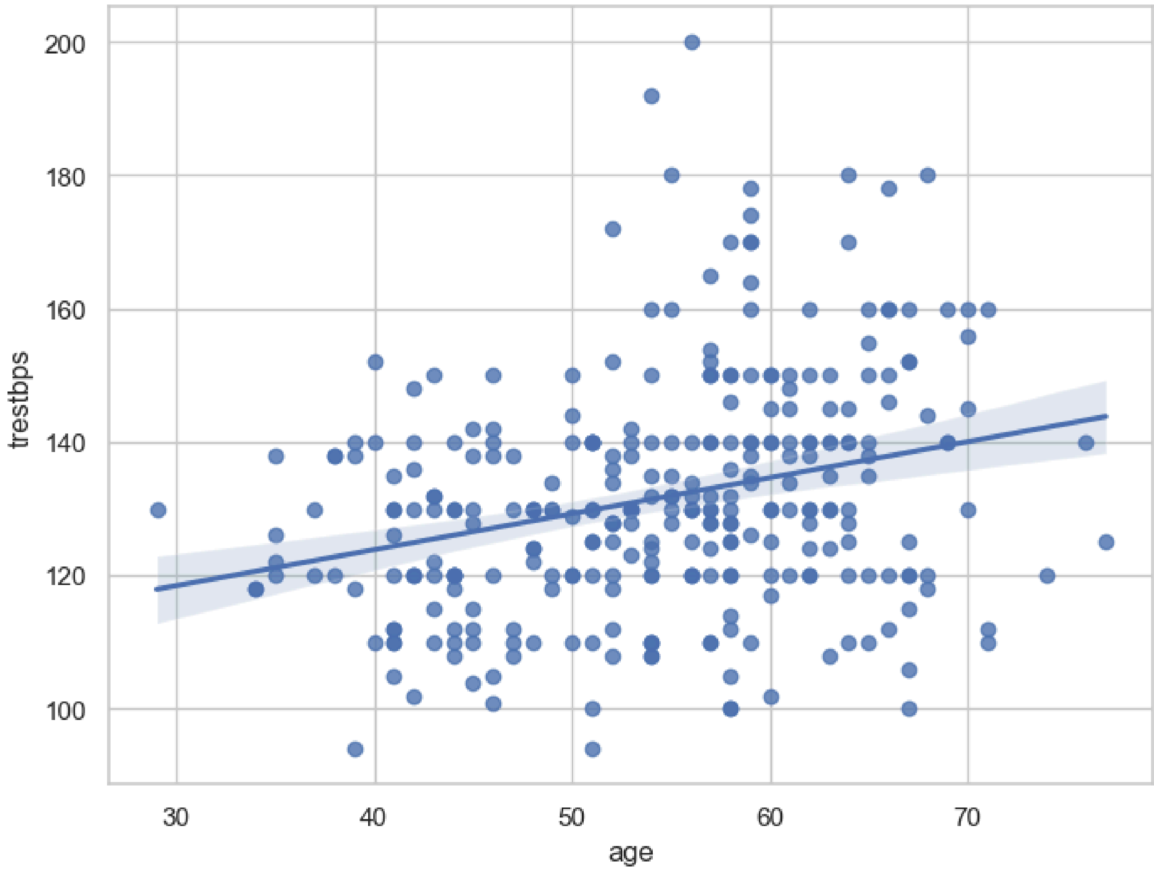
```
ax=plt.subplots(figsize=(8,6))
ax=sns.scatterplot(x='age', y='trestbps', data=df)
plt.show()
```



- The above scatter plot shows that there is no correlation between `age` and `trestbps` variable.

In [280…

```
ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="age", y="trestbps", data=df)
plt.show()
```

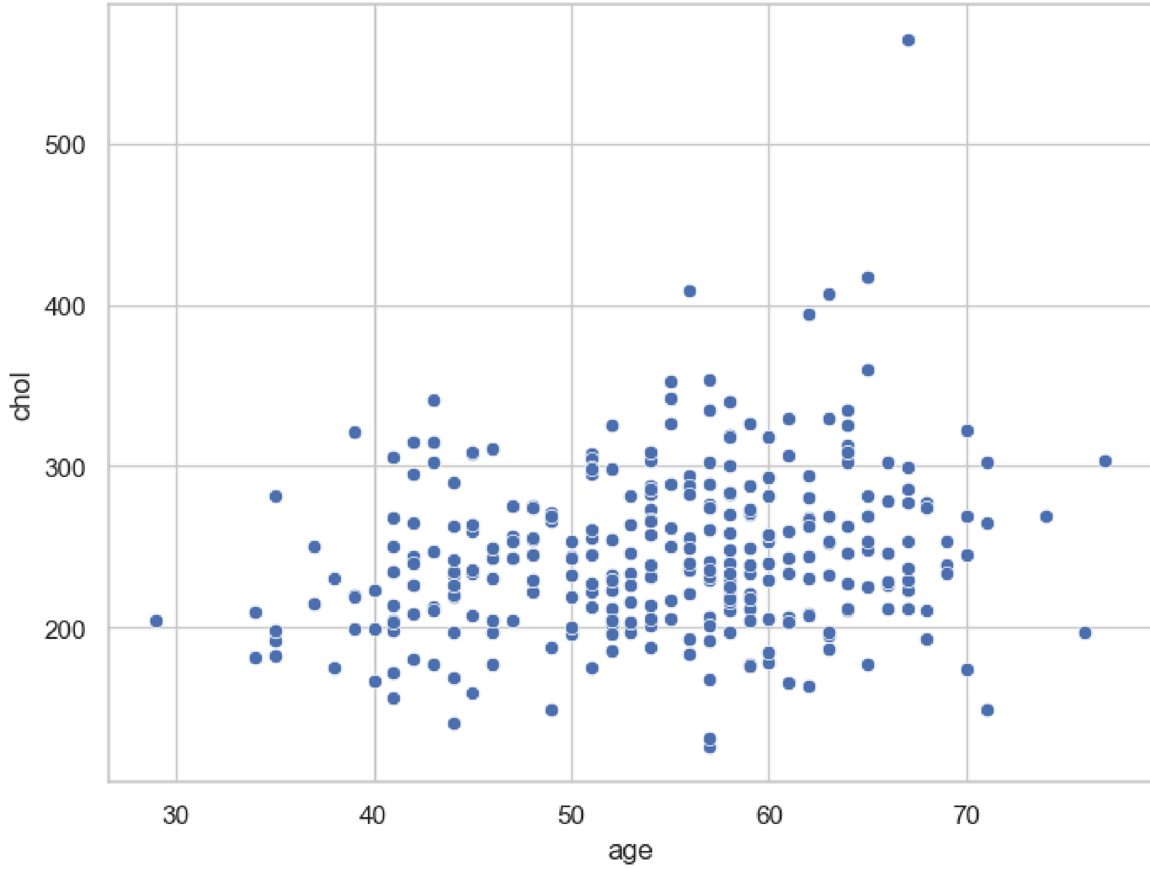


- The above line shows that linear regression model is not good fit to the data.

Analyze `age` and `chol` variable

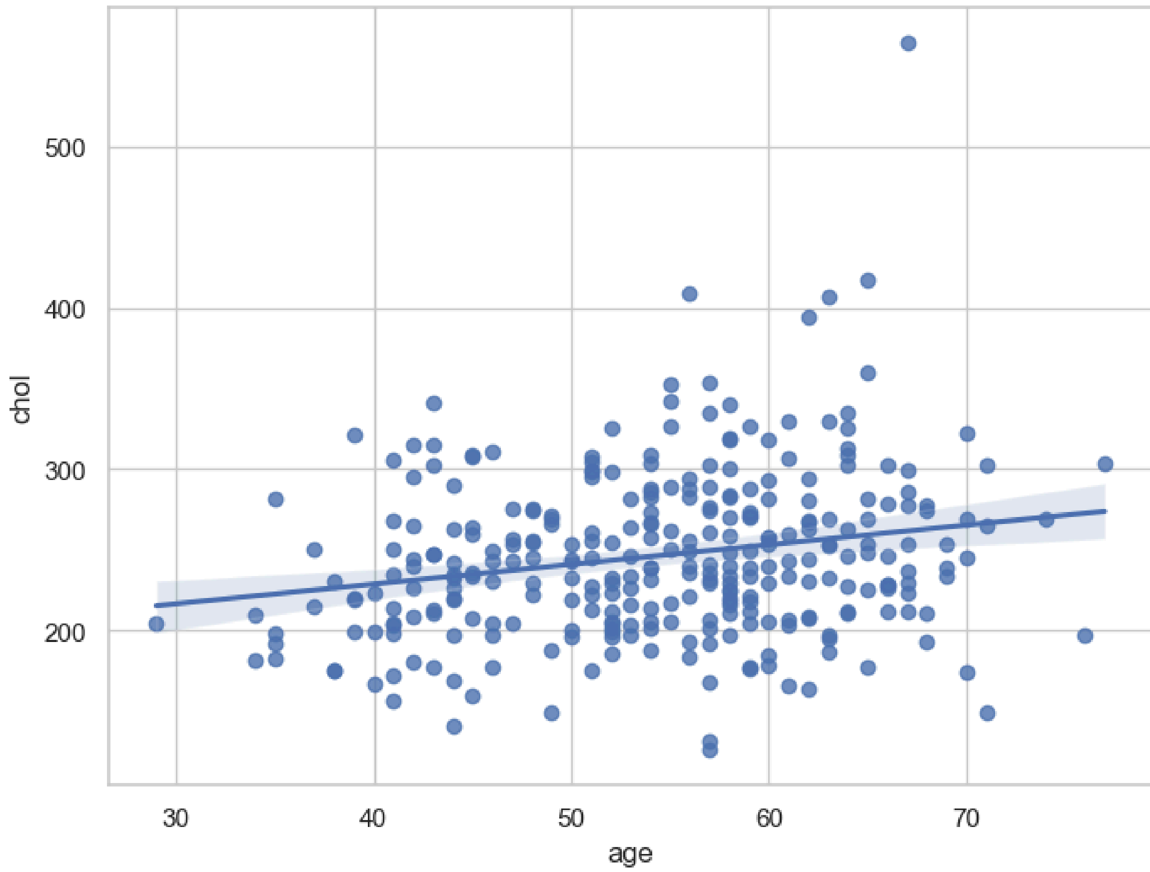
In [282…

```
ax = plt.subplots(figsize=(8, 6))
ax = sns.scatterplot(x="age", y="chol", data=df)
plt.show()
```



In [284…

```
ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="age", y="chol", data=df)
plt.show()
```

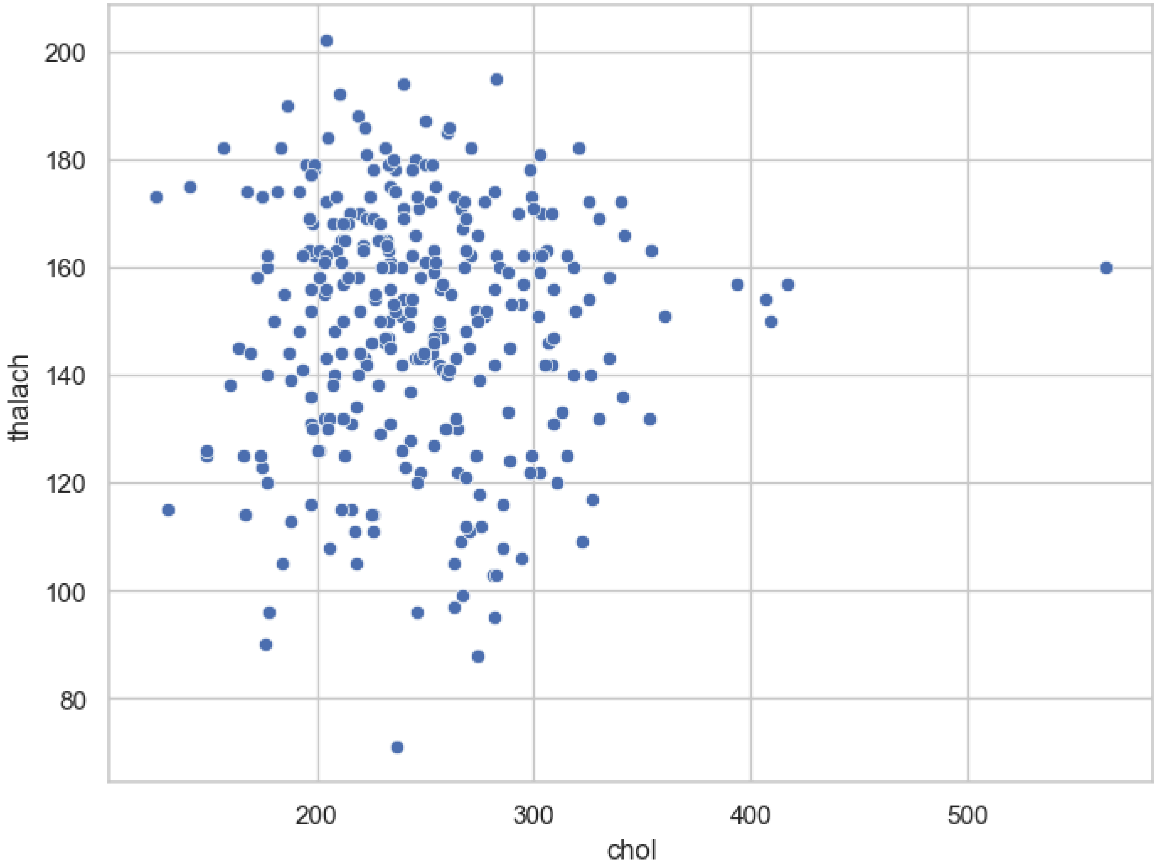


- The above plot confirms that there is a slighly positive correlation between `age` and `chol` variables.

Analyze `chol` and `thalach` variable

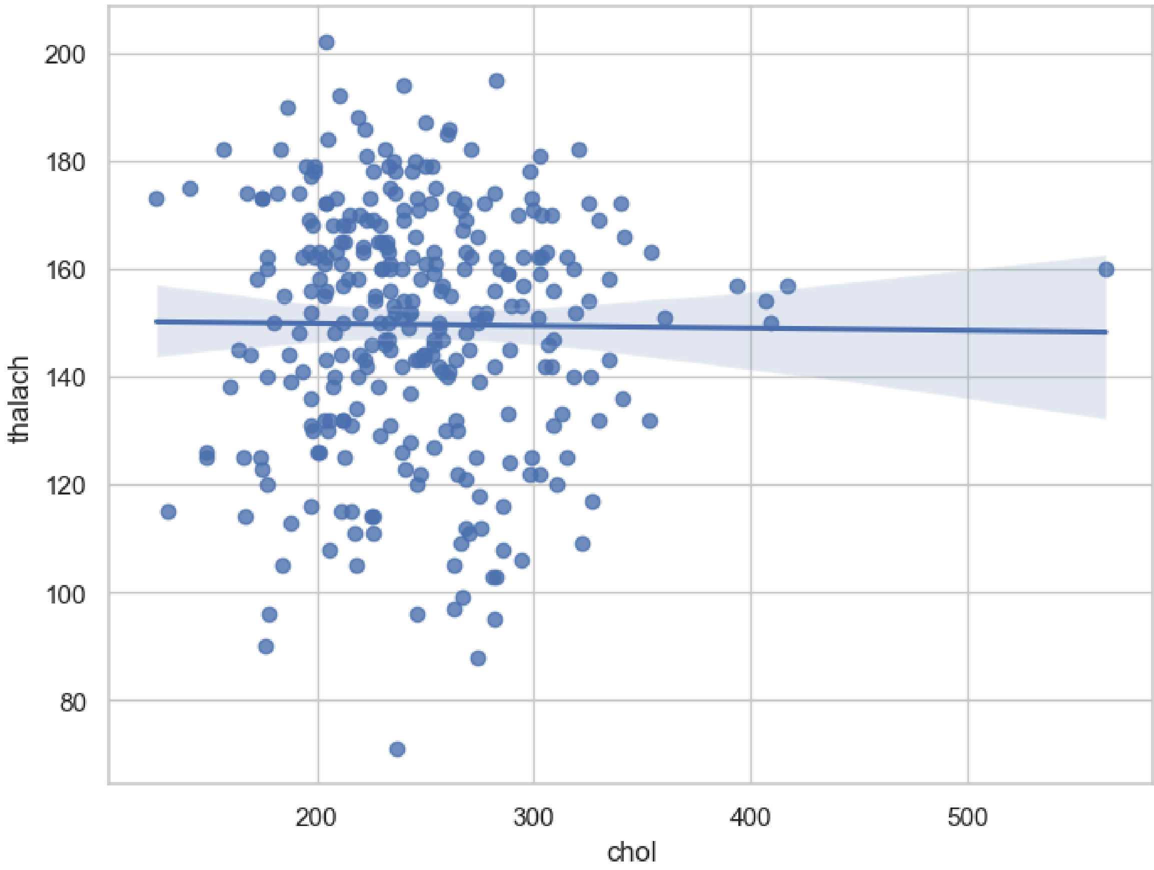
In [286…

```
ax = plt.subplots(figsize=(8, 6))
ax = sns.scatterplot(x="chol", y = "thalach", data=df)
plt.show()
```



In [288…

```
ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="chol", y="thalach", data=df)
plt.show()
```



- The above plot shows that there is no correlation between `chol` and `thalach` variable.

Dealing with missing values

In [290…
```
df.isnull().sum()
```

Out[290…
```
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
```

Check with ASSERT statement

- 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.

In [292...
```
assert pd.notnull(df).all().all()
```

In [294...
```
assert (df >= 0).all().all()
```

- The above two commands do not throw any error. Hence, it is confirmed that there are no missing or negative values in the dataset.
- All the values are greater than or equal to zero.

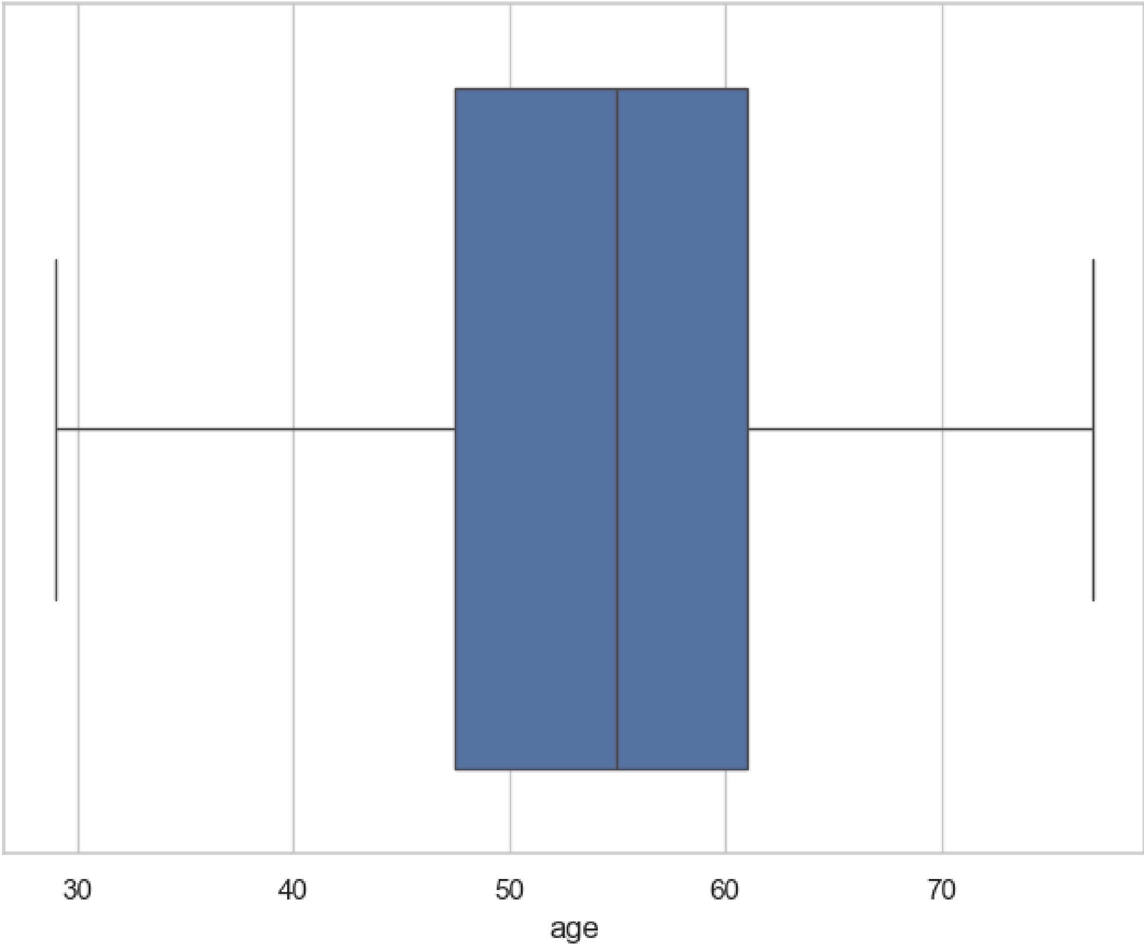
In [296...
```
df.age.describe()
```

Out[296...
```
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
```

- I will make boxplots to visualise outliers in the continuous numerical variables : -

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

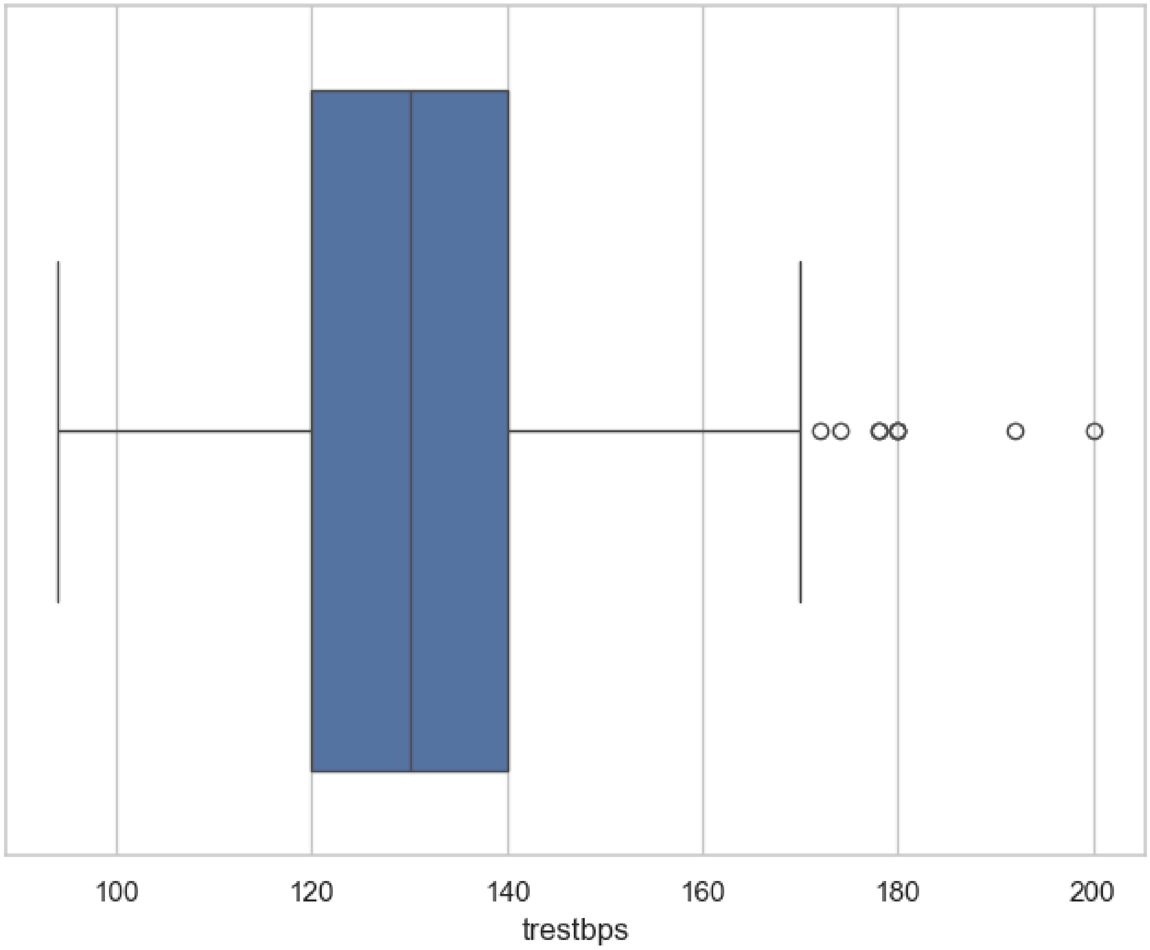
In [306...
```
ax=plt.subplots(figsize=(8,6))
sns.boxplot(x=df.age)
plt.show()
```



In [308…
```
df['trestbps'].describe()
```

Out[308…
```
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 [310…
```
ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["trestbps"])
plt.show()
```



In [312…

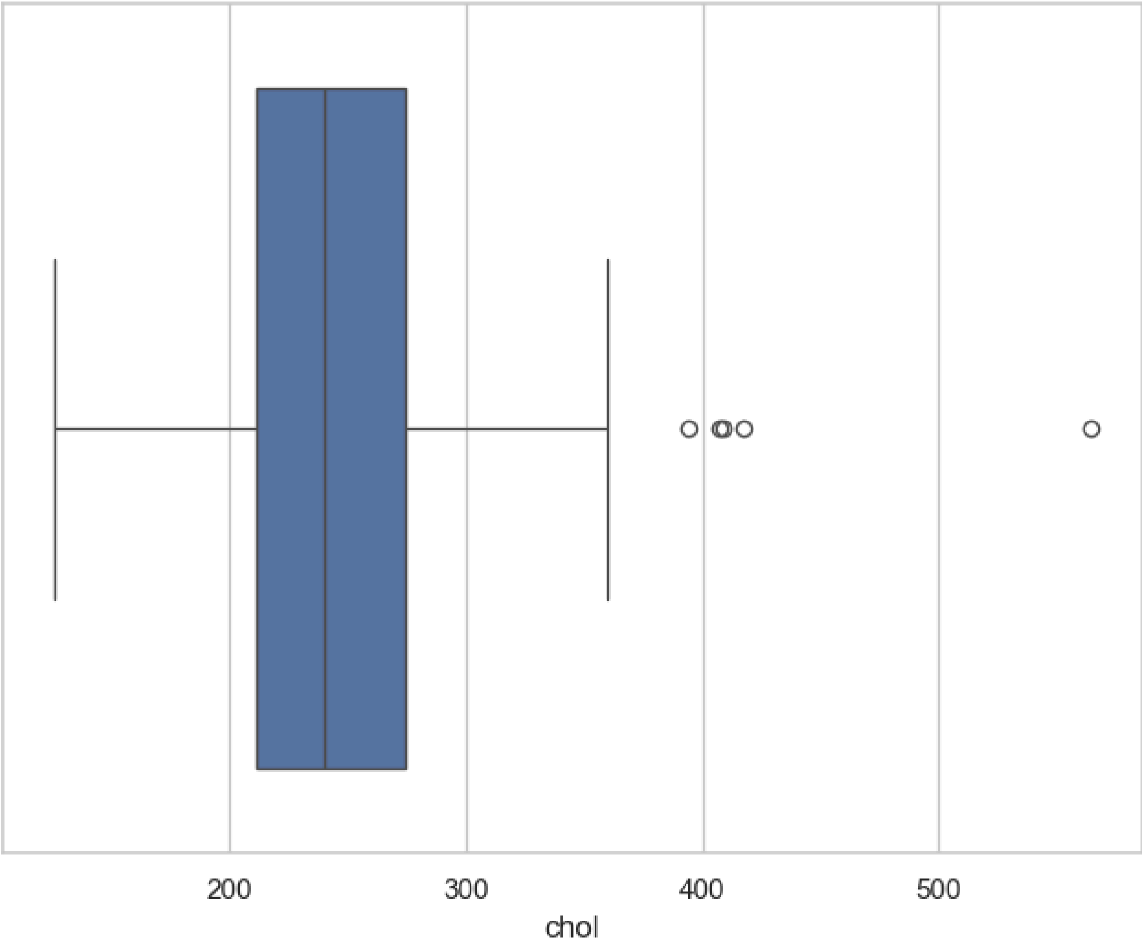
```
df['chol'].describe()
```

Out[312…

```
count    303.000000
mean     246.264026
std       51.830751
min      126.000000
25%      211.000000
50%      240.000000
75%      274.500000
max      564.000000
Name: chol, dtype: float64
```

In [314…

```
ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["chol"])
plt.show()
```



In [316…

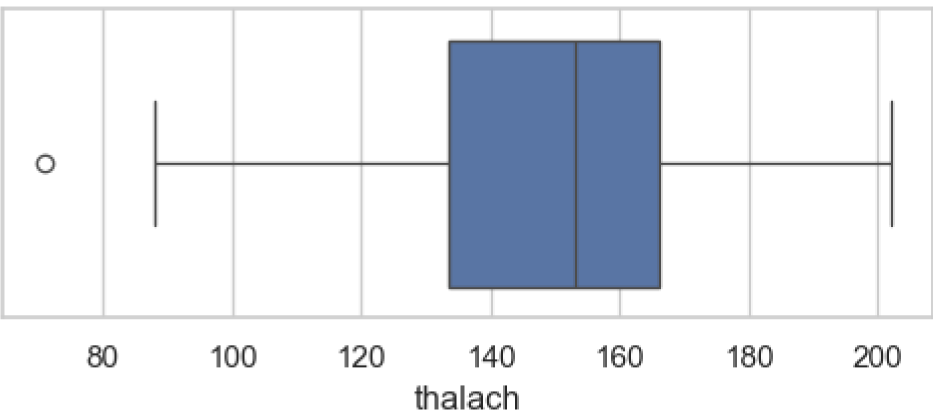
```
df['thalach'].describe()
```

Out[316…

```
count    303.000000
mean     149.646865
std       22.905161
min       71.000000
25%      133.500000
50%      153.000000
75%      166.000000
max      202.000000
Name: thalach, dtype: float64
```

In [320…

```
ax = plt.subplots(figsize=(6, 2))
sns.boxplot(x=df["thalach"])
plt.show()
```



In [322…

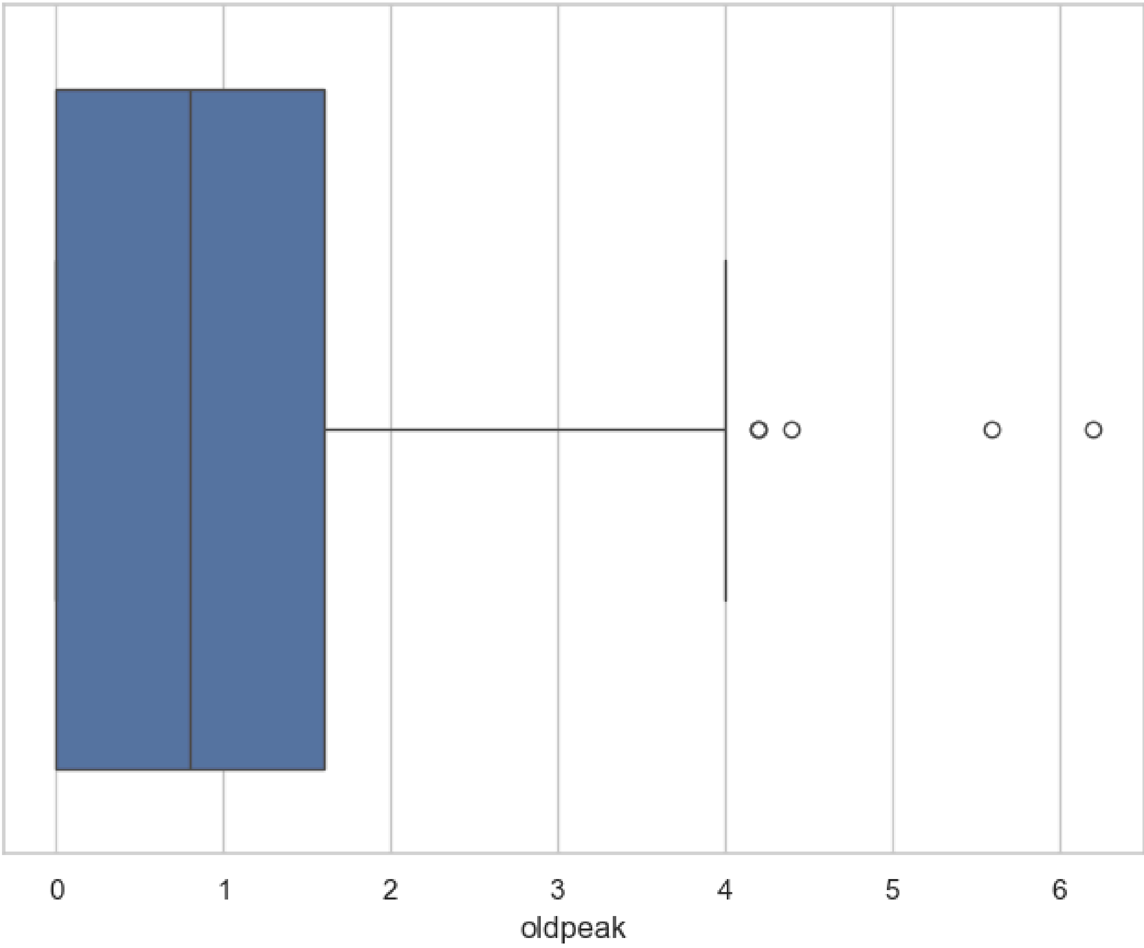
```
df['oldpeak'].describe()
```

Out[322…

```
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 [324…

```
ax = plt.subplots(figsize=(8, 6))
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.

Conclusion:

- This project demonstrates a comprehensive exploratory data analysis (EDA) and visualization of the Heart Disease UCI dataset using Python. Through univariate, bivariate, and multivariate analysis, key insights into the factors contributing to heart disease were uncovered. The analysis revealed significant relationships between features such as chest pain type, maximum heart rate achieved, and the presence of heart disease. Visualizations, including count plots, bar plots, and heatmaps, were effectively used to present the findings in an intuitive and actionable manner.
- This work serves as a foundation for further predictive modeling and machine learning applications, such as building classification models to predict heart disease. It also underscores the importance of thorough data exploration and visualization in understanding and solving real-world problems. Overall, this project exemplifies my ability to analyze, visualize, and interpret data, making it a valuable addition to my portfolio and a strong demonstration of my skills for roles in data analysis, data science, and healthcare analytics.