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.

So, in this kernel, I have conducted **Exploratory Data Analysis** or **EDA** of the heart disease dataset. **Exploratory Data Analysis** or **EDA** is a critical first step in analyzing a new dataset. The primary objective of EDA is to analyze the data for distribution, outliers and anomalies in the dataset. It enable us to direct specific testing of the hypothesis. It includes analysing the data to find the distribution of data, its main characteristics, identifying patterns and visualizations. It also provides tools for hypothesis generation by visualizing and understanding the data through graphical representation.

I hope you learn and enjoy this kernel.

Table of Contents

The table of contents for this project are as follows: -

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

1. Introduction to EDA

Back to Table of Contents

Several questions come to mind when we come across a new dataset. The below list shed light on some of these questions:-

• What is the distribution of the dataset?

• Are there any missing numerical values, outliers or anomalies in the dataset?

• What are the underlying assumptions in the dataset?

• Whether there exists relationships between variables in the dataset?

• How to be sure that our dataset is ready for input in a machine learning algorithm?

• How to select the most suitable algorithm for a given dataset?

So, how do we get answer to the above questions?

The answer is **Exploratory Data Analysis**. It enable us to answer all of the above questions.

2. Objectives of EDA

Back to Table of Contents

The objectives of the EDA are as follows:-

i. To get an overview of the distribution of the dataset.

ii. Check for missing numerical values, outliers or other anomalies in the dataset.

iii.Discover patterns and relationships between variables in the dataset.

iv. Check the underlying assumptions in the dataset.

3. Types of EDA

There are `seven` types of exploratory data analysis (EDA)based on the above cross-classification methods

1.variable identificatiom

2.Univariate non-graphical EDA

3.bivarient analysis

4.missing value treatment

5.imputation technique

6.variable creation

7.outlier detection

Import libraries

In [1]:
```
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
%matplotlib inline
```

import warnings

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

Import dataset

In [3]:
```
df = pd.read_csv(r'C:\Users\ADMIN\Desktop\DATA SCIENCE NOTES\NOVEMBER MONTH\27TH NOV\25th, 26th- Advanced EDA project
```

In [4]:
```
df
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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

303 rows × 14 columns

Exploratory Data Analysis

In [5]:
```
print('the shape of the dataset : ',df.shape)
```

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

preview the dataset

In [6]:
```
df.head()
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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

Summary of dataset

In [7]:
```
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
```

In [8]:
```
df.dtypes
```

Out[8]:
```
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
```

Statistical properties of dataset

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

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
count	303.000000	303.000000	303.000000	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	149.646865	0.326733	1.039604
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000

View the column names

In [10]:
```
df.columns
```

Out[10]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

univarient analysis

Analysis of `target` feature variable

- 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).
- So, in this section, I will analyze the `target` variable.

check th enumber of unique values in `target` variable

In [11]:
```
df['target'].unique()
```

Out[11]:
```
array([1, 0], dtype=int64)
```

In [12]:
```
df['target'].nunique()
```

Out[12]:
```
2
```

Frequency distribution of target variable

In [13]:
```
df['target'].value_counts()
```

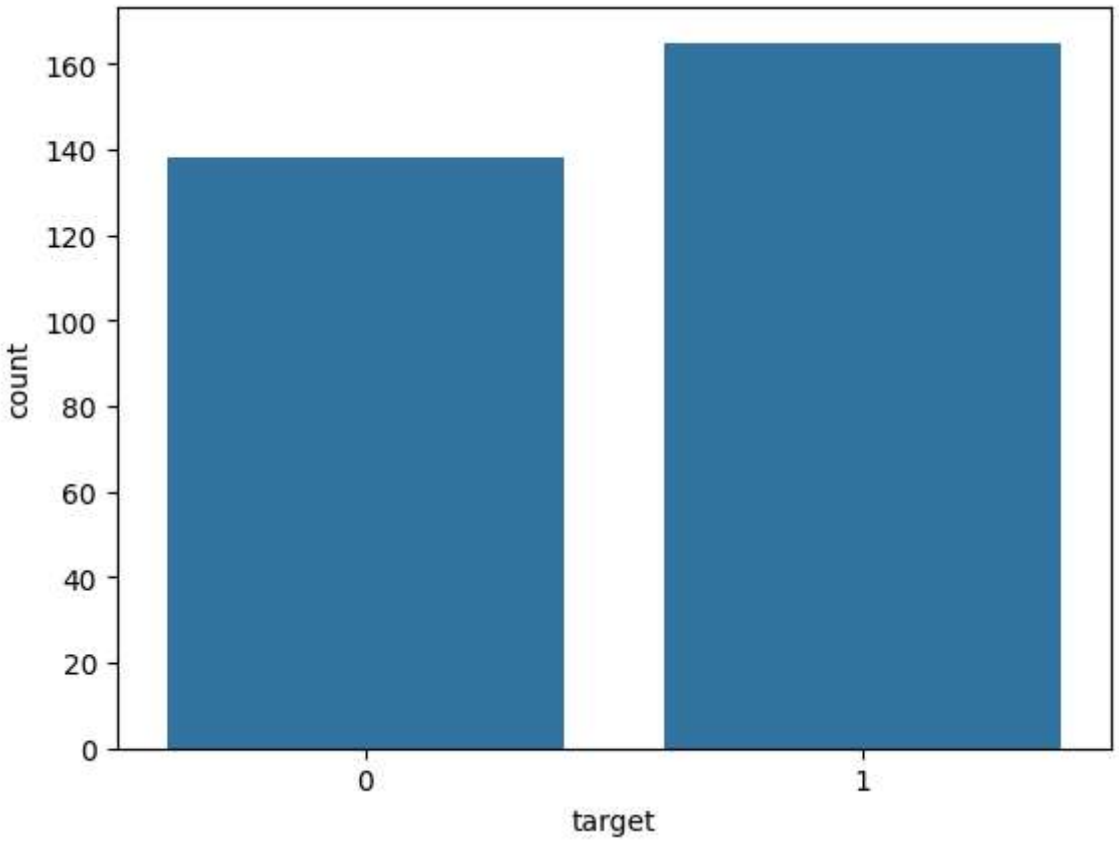
Out[13]:
```
target
1    165
0    138
Name: count, dtype: int64
```

Comment

- `1` means presence of heart disease. So , 165 patients suffering from heart disease
- `0` means no heart disease , 138 dont have any heart disease

In [14]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(6.4, 4.8))
sns.countplot(data=df, x='target')
plt.show()
```



Frequency distribution of `target` variable wrt sex

In [15]:
```
df.groupby('sex')['target'].value_counts()
```

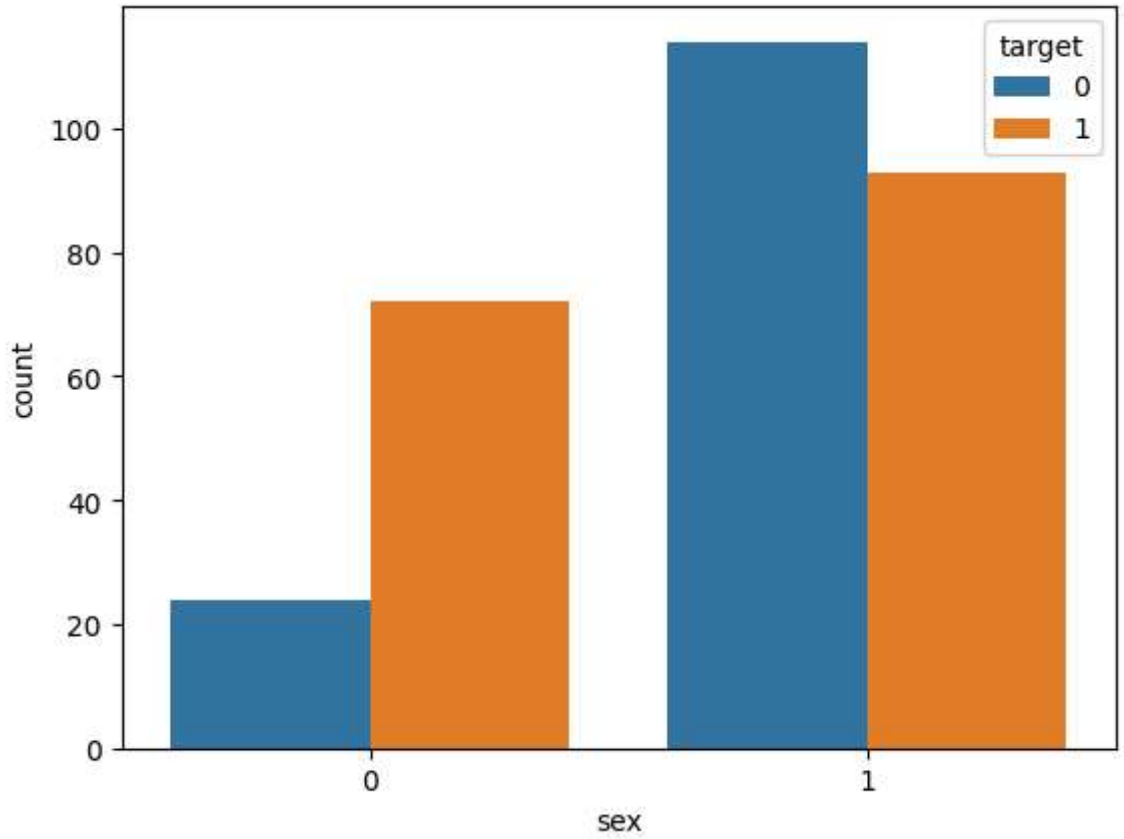
Out[15]:
```
sex  target
0    1          72
     0          24
1    0         114
     1          93
Name: count, dtype: int64
```

comment

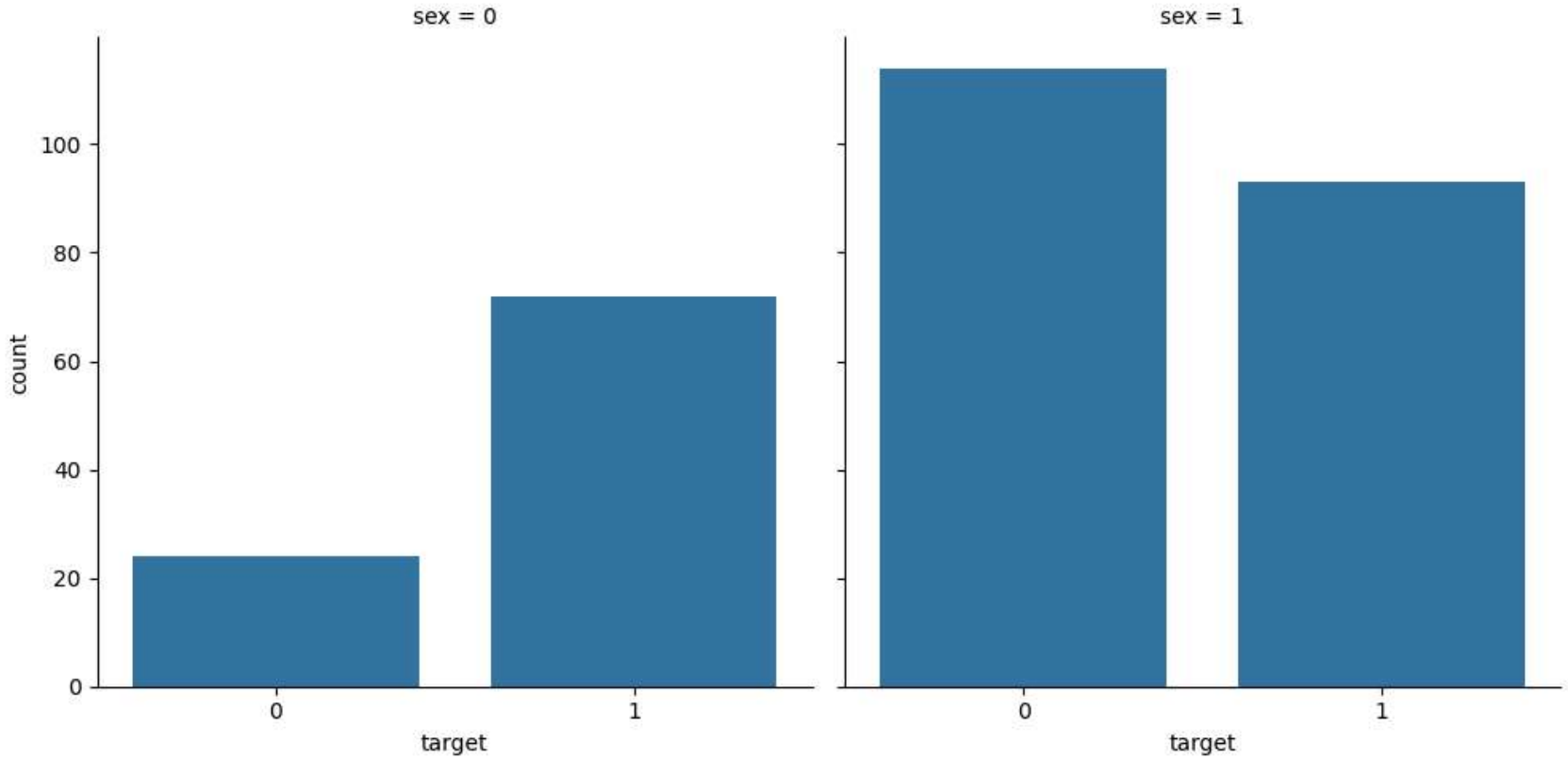
- `sex` (1 = male , 0 = female):
- `target` (1 = have disease , 0 = no disease)
- out of 165 males -- 72 have disease and 24 dont have disease
- out of 138 females -- 93 have diseas and 114 dont have disease

we can visualize the value counts of the sex variable wrt target

In [16]:
```
plt.figure(figsize=(6.4, 4.8))
sns.countplot(data=df, x = 'sex' , hue='target')
plt.show()
```

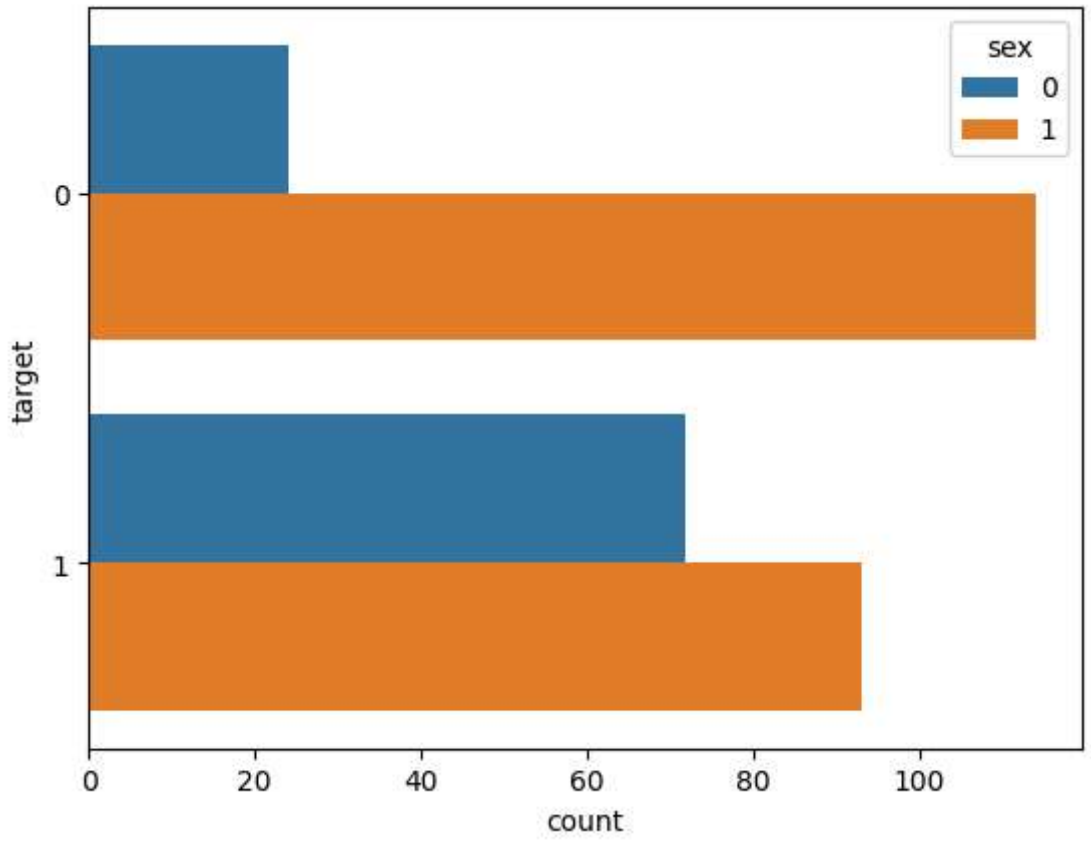


In [17]:
```
ax = sns.catplot(data=df,x='target',col='sex',kind='count',height=5,aspect=1)
```



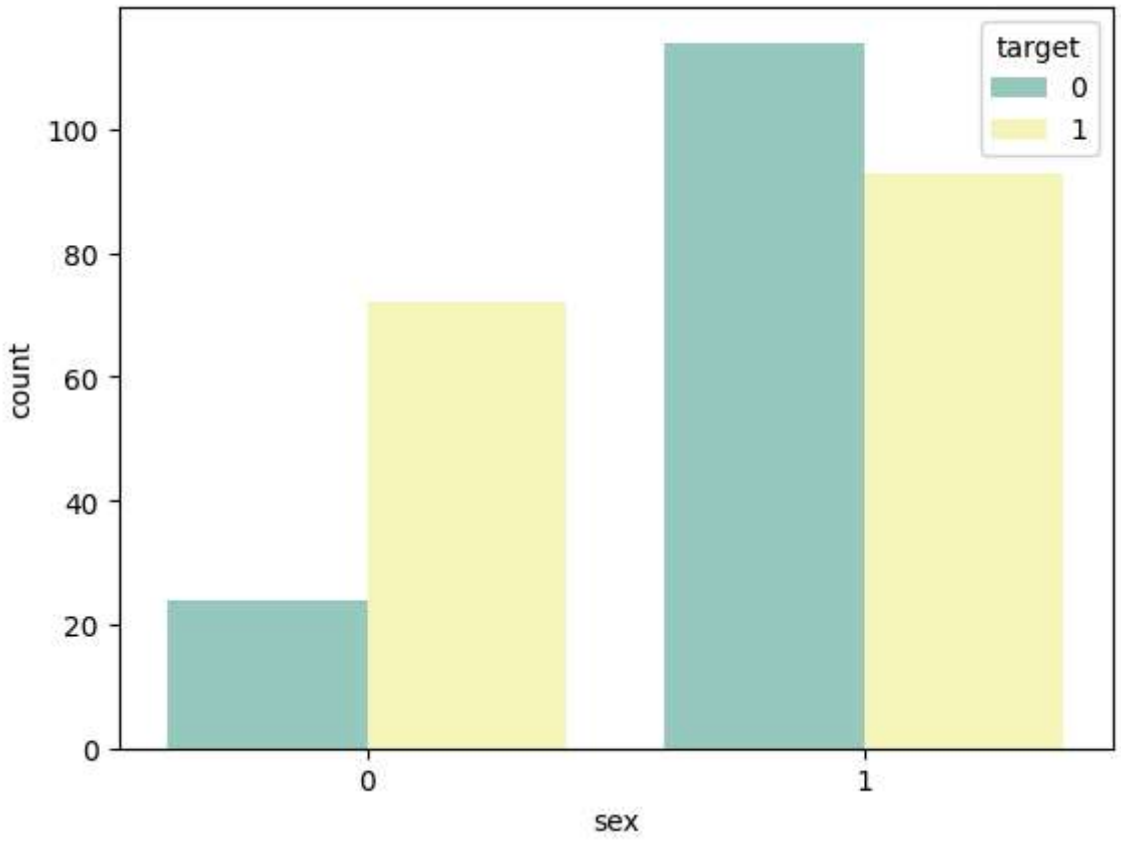
we can plot the bars horizontally

In [18]:
```
plt.figure(figsize=(6.4, 4.8))
sns.countplot(data=df, hue = 'sex' , y='target')
plt.show()
```

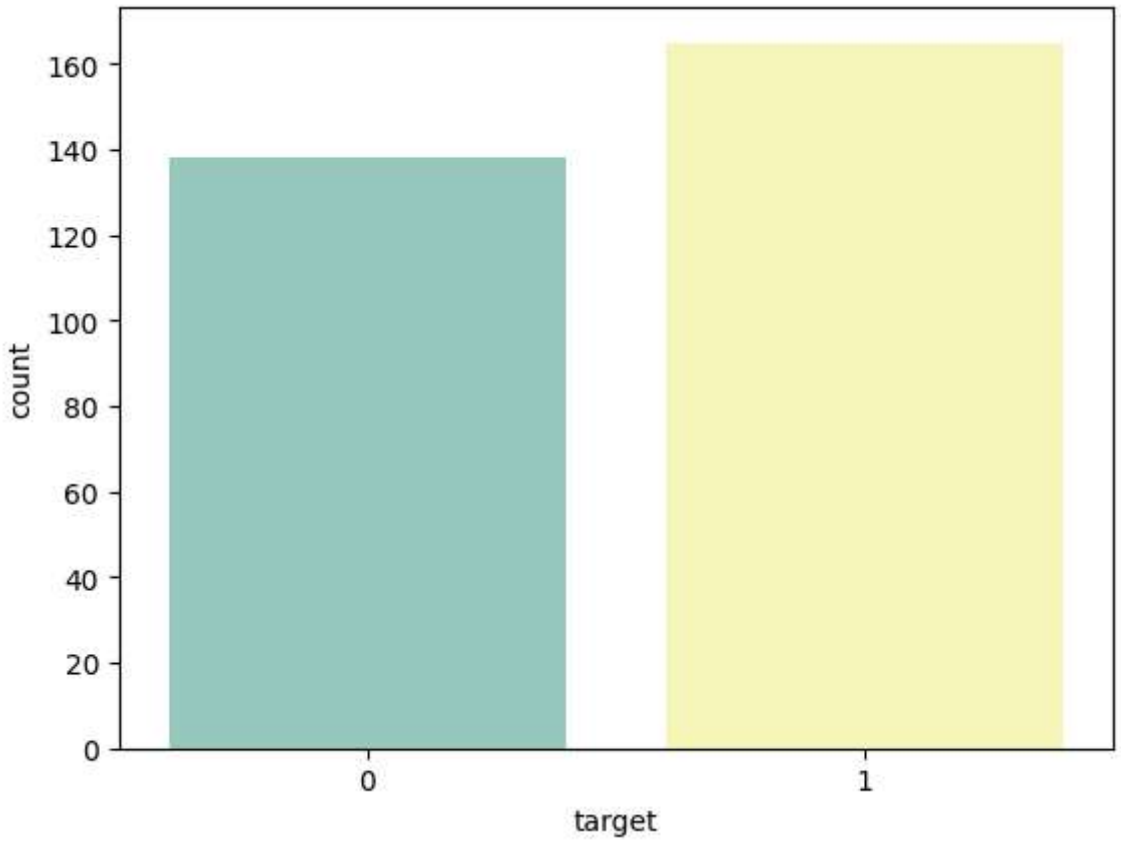


we can use a different color palette

In [19]:
```
plt.figure(figsize=(6.4, 4.8))
sns.countplot(data=df, x = 'sex' , hue='target',palette = 'Set3')
plt.show()
```

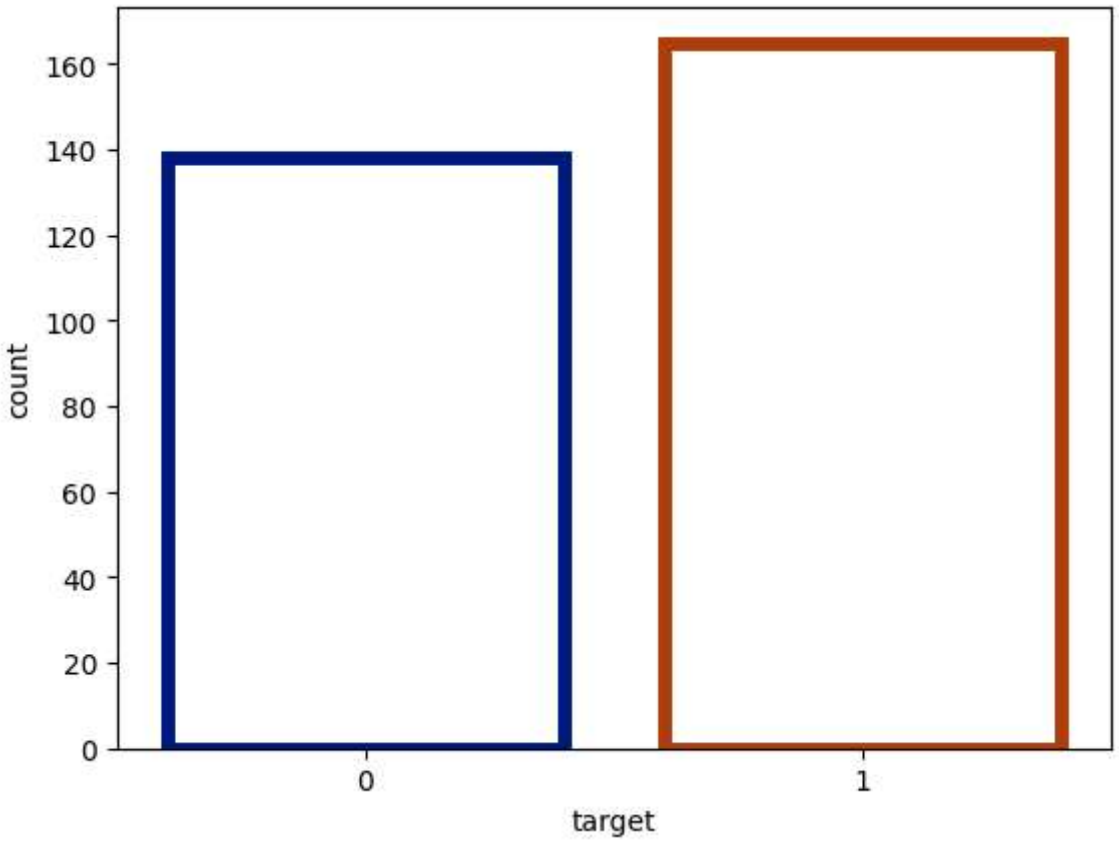


In [20]:
```
plt.figure(figsize=(6.4, 4.8))
sns.countplot(data=df, x ='target',palette = 'Set3')
plt.show()
```



we can use `plt.bar` keyword argument for a different look

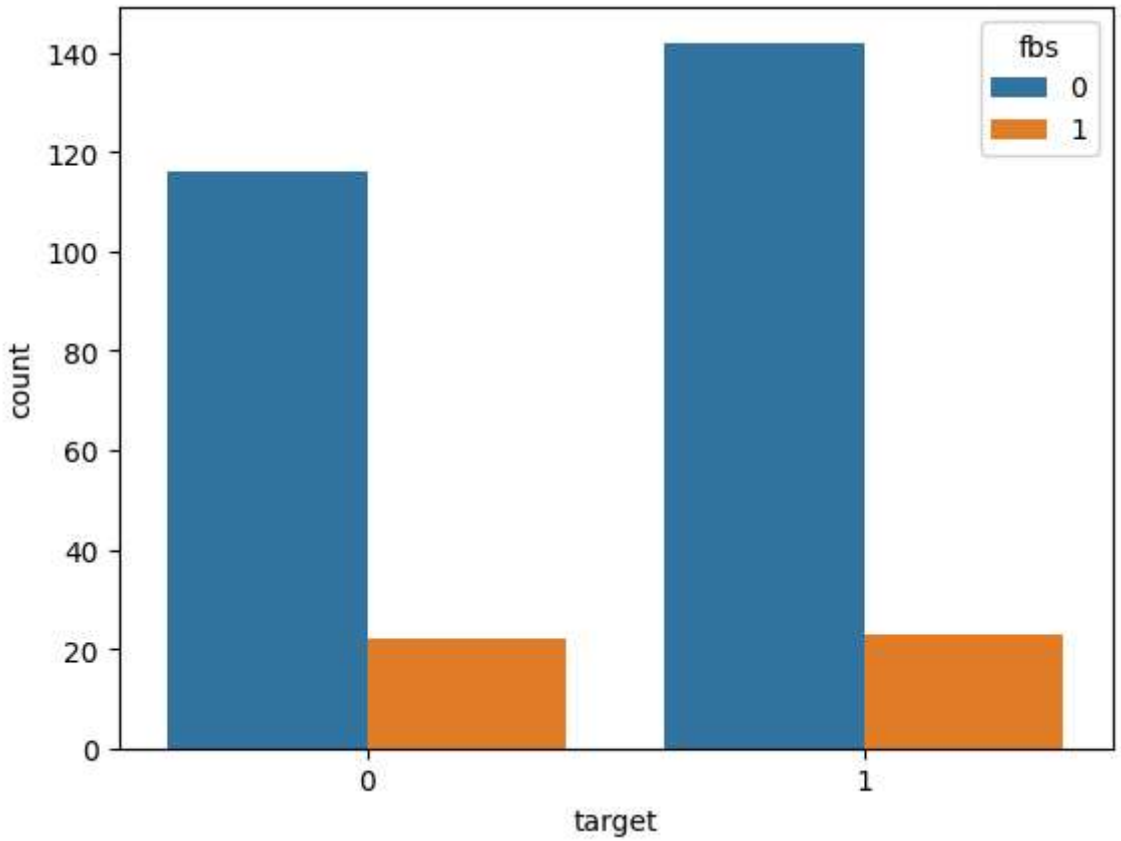
In [21]:
```
plt.figure(figsize=(6.4, 4.8))
sns.countplot(data=df, x ='target',facecolor = (0, 0, 0, 0), linewidth = 5 , edgecolor = sns.color_palette('dark',3))
plt.show()
```



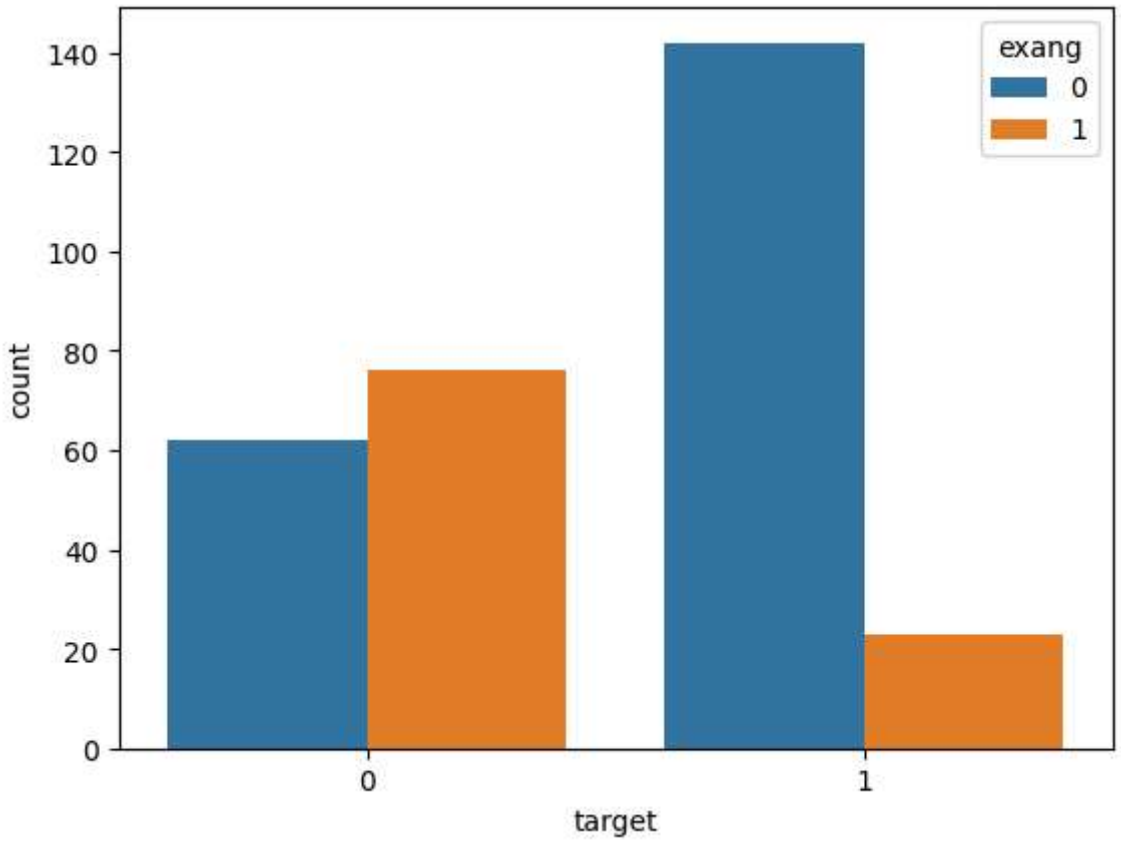
comment

- I have visualize the `target` values distributions wrt `sex`
- now we have to compare the `target` variable wrt `fbs` and `exang`

In [22]:
```
plt.figure(figsize=(6.4, 4.8))
sns.countplot(data=df, x ='target',hue= 'fbs' )
plt.show()
```



In [23]:
```
plt.figure(figsize=(6.4, 4.8))
sns.countplot(data=df, x ='target',hue= 'exang' )
plt.show()
```



Bivariant Analysis

Estimate correlation coefficients

- Now we will compute the standard correlation coefficient between every pair of attributes

In [24]:
```
correlation = df.corr()
correlation
```

Out[24]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877

In [25]:
```
correlation['target'].sort_values(ascending=False)
```

Out[25]:
```
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

- correlation coefficient ranges from `-1 to +1`
- when its close to +ve then it as strong positive correlation
- from above code we observe that no variable which have strong positive correlation with `target`
- No correlation with `target` and `fbs`
- we can see that `target = 1.000000` `cp= 0.43379` `thalac= 0.4217` have slightly positive correlation 41

Analysis of `target` and `cp` variables

- cp means chest pain

In [26]:
```
df['cp'].unique()
```

Out[26]:
```
array([3, 2, 1, 0], dtype=int64)
```

In [27]:
```
df['cp'].nunique()
```

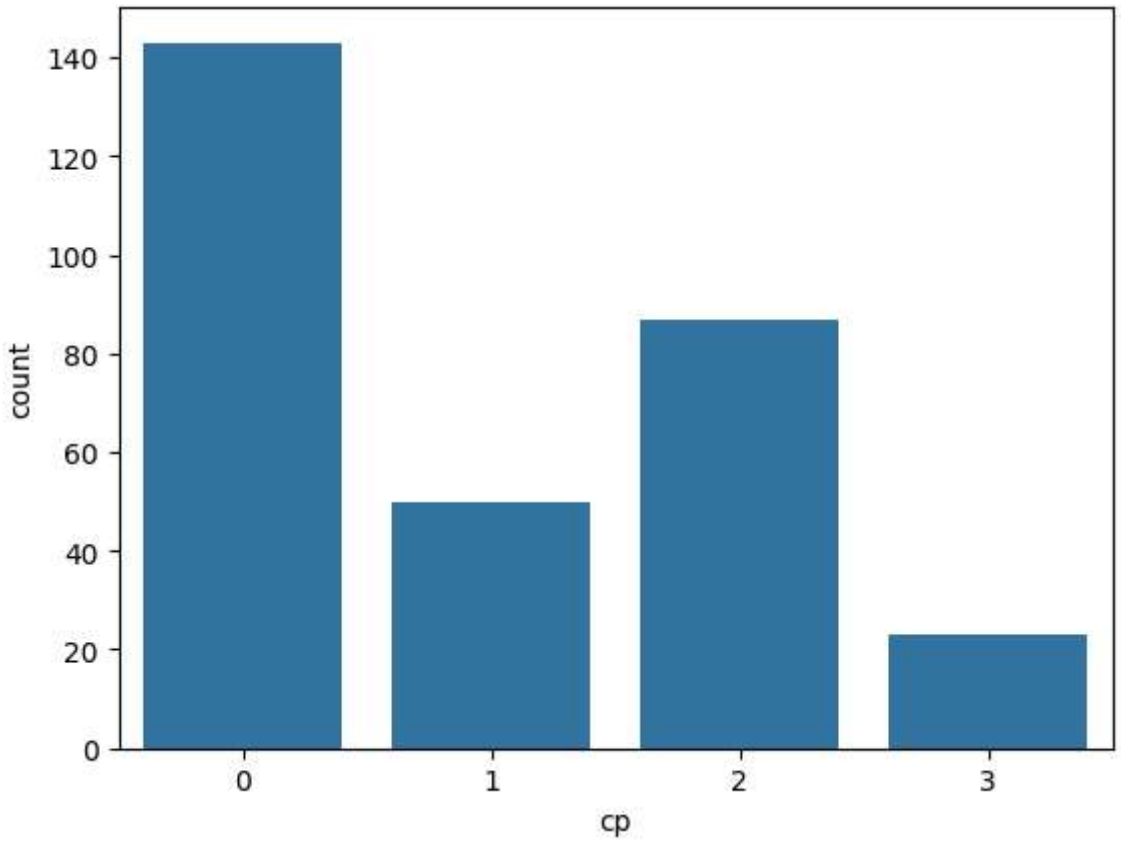
Out[27]:
```
4
```

In [28]:
```
df['cp'].value_counts()
```

Out[28]:
```
cp
0    143
2     87
1     50
3     23
Name: count, dtype: int64
```

visualize the frequency distribution of `cp` variable

In [29]:
```
plt.figure(figsize=(6.4, 4.8))
sns.countplot(data=df, x ='cp' )
plt.show()
```



Frequency distribution of target variable wrt cp

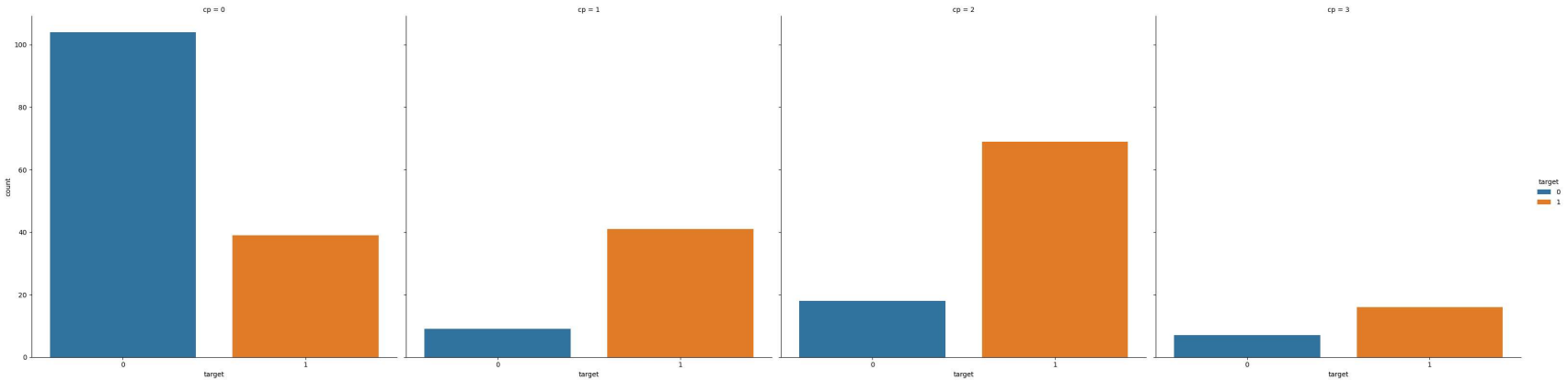
In [30]:
```
df.groupby('cp')['target'].value_counts()
```

Out[30]:
```
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
```

comment

- from above analysis gives `target` with groupby `cp`
- cp - chest pain type
- 0 Typical angina
- 1 Atypical angina
- 2 Non-anginal pain
- 3 Asymptomatic (no chest pain)slient heart disease
- cp - 0, 1 , 2 , 3 = `39` , `41` , `69` , `16` have heart disease

In [31]:
```
ax = sns.catplot(x="target", col="cp", data=df, kind="count", height=8, aspect=1,hue = 'target')
```



Analysis of `target` and `thalach` variable

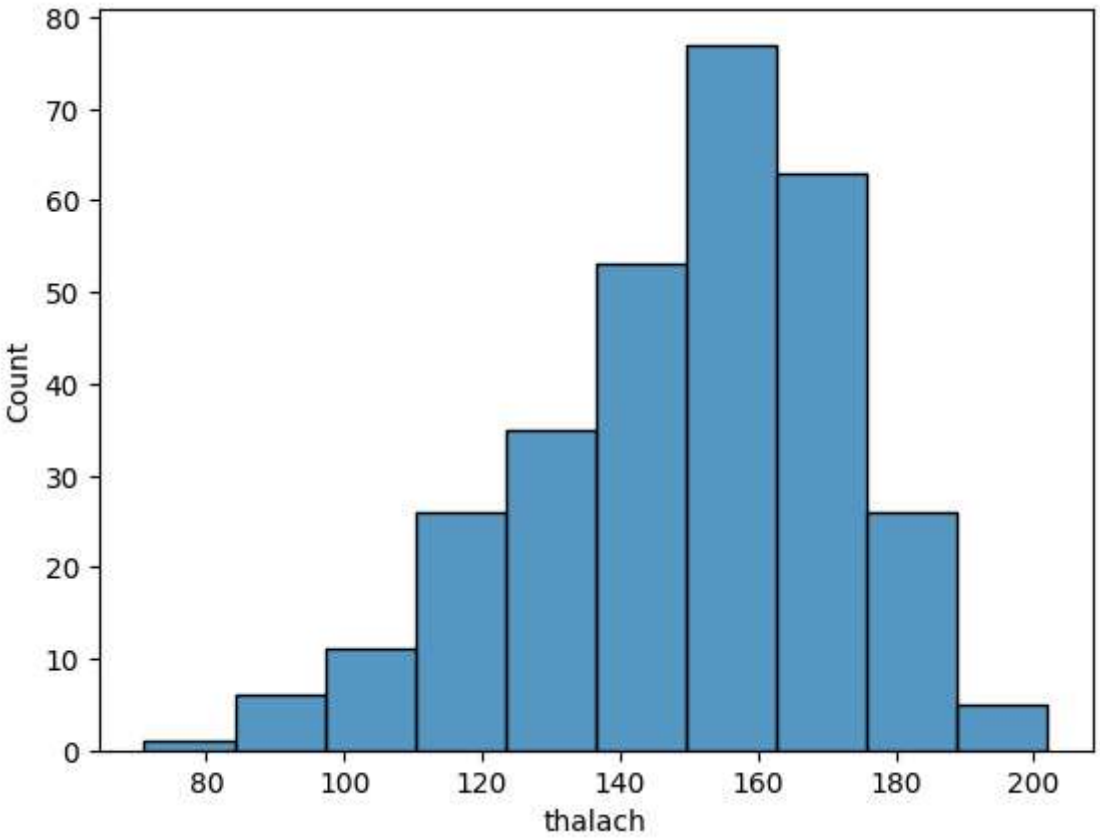
- `thalach` stands for maximum heart rate achieved

In [32]:
```
df['thalach'].nunique()
```

Out[32]: 91

Visualize the frequency distribution of `thalach` variable

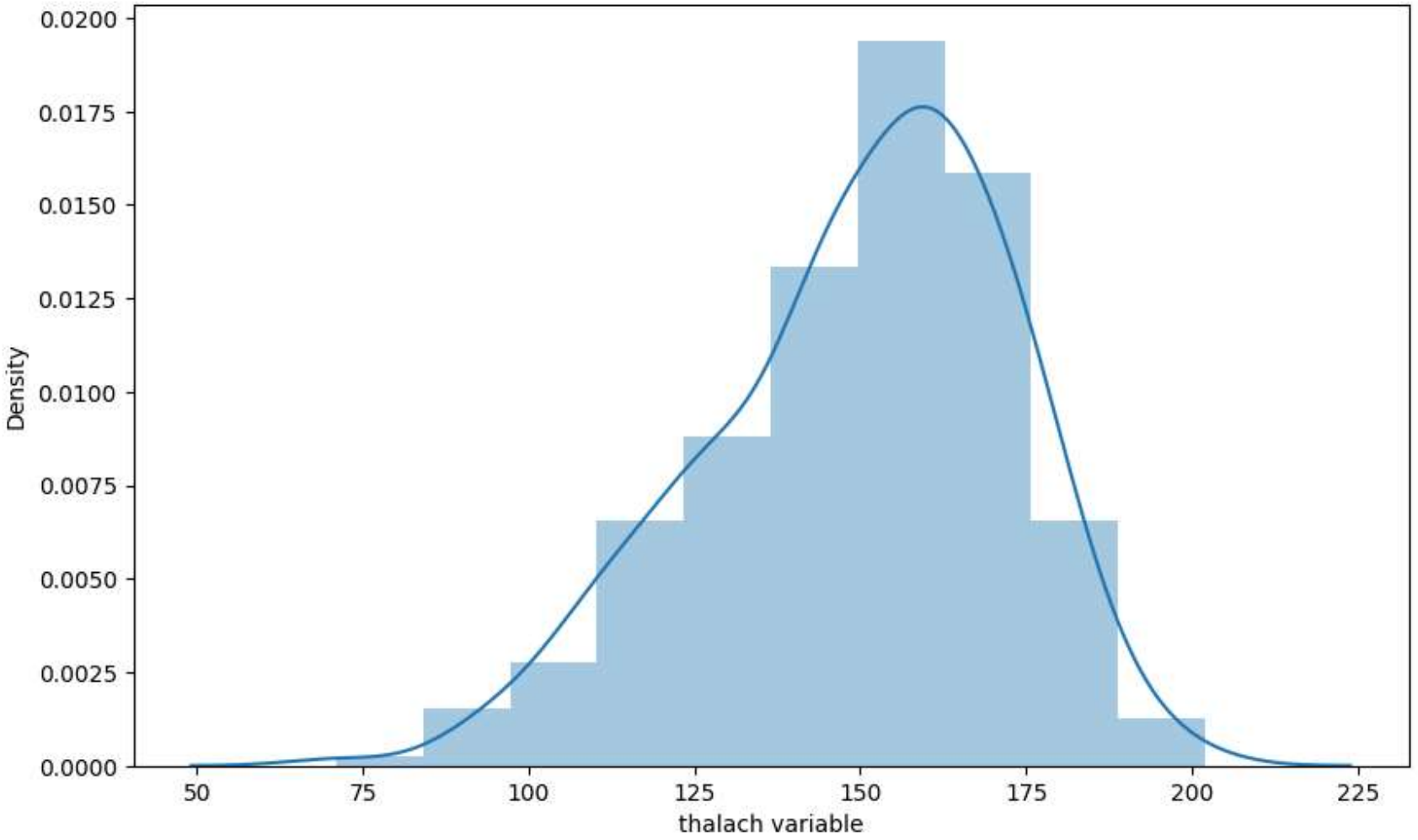
In [33]:
```
plt.figure(figsize=(6.4, 4.8))
sns.histplot(data = df ,x = 'thalach',bins=10)
plt.show()
```



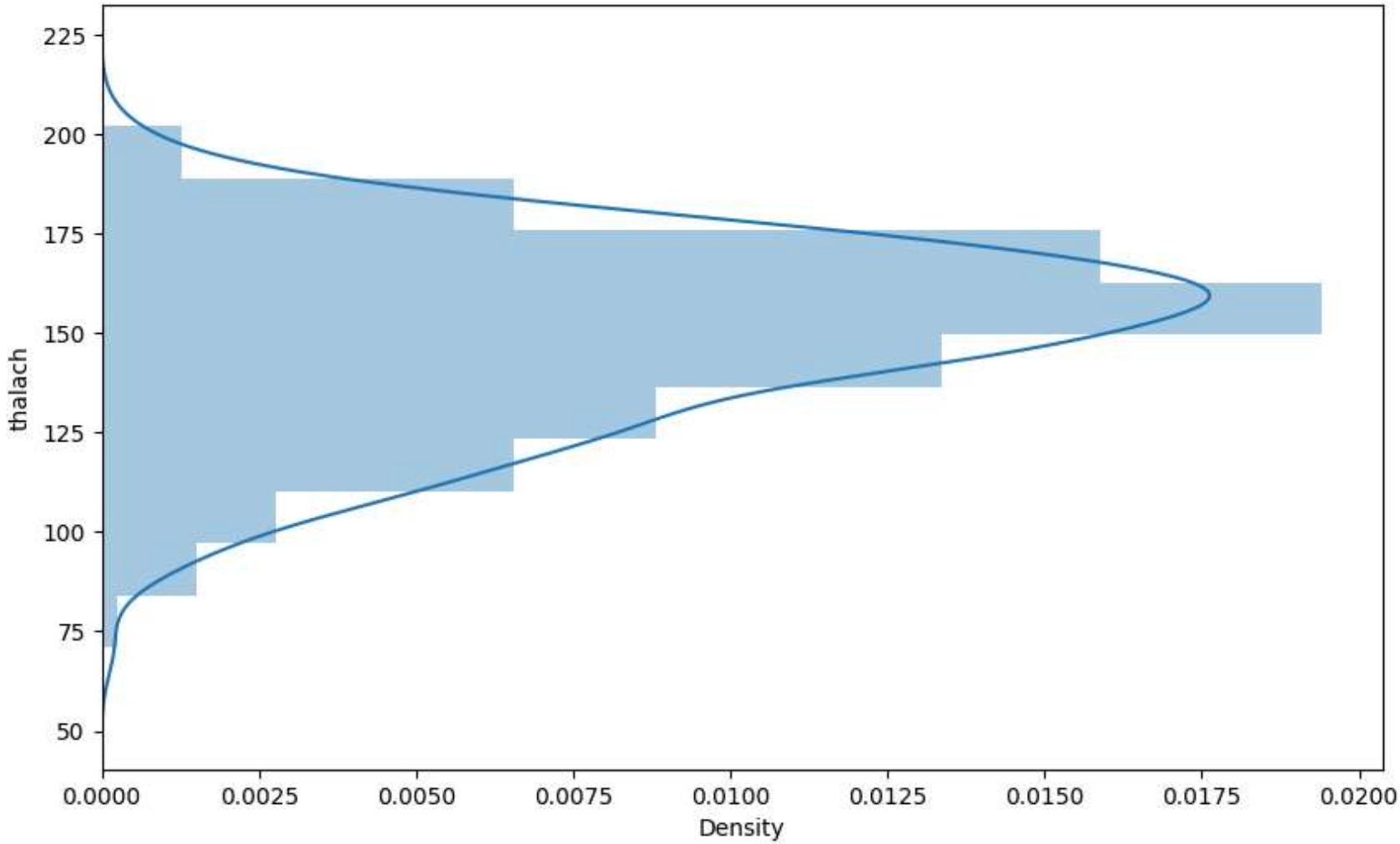
comment

- we can see that the `thalach` variable is slightly negative skewed

In [34]:
```
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 [35]:
```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
ax = sns.distplot(x, bins=10,vertical = True)
plt.show()
```

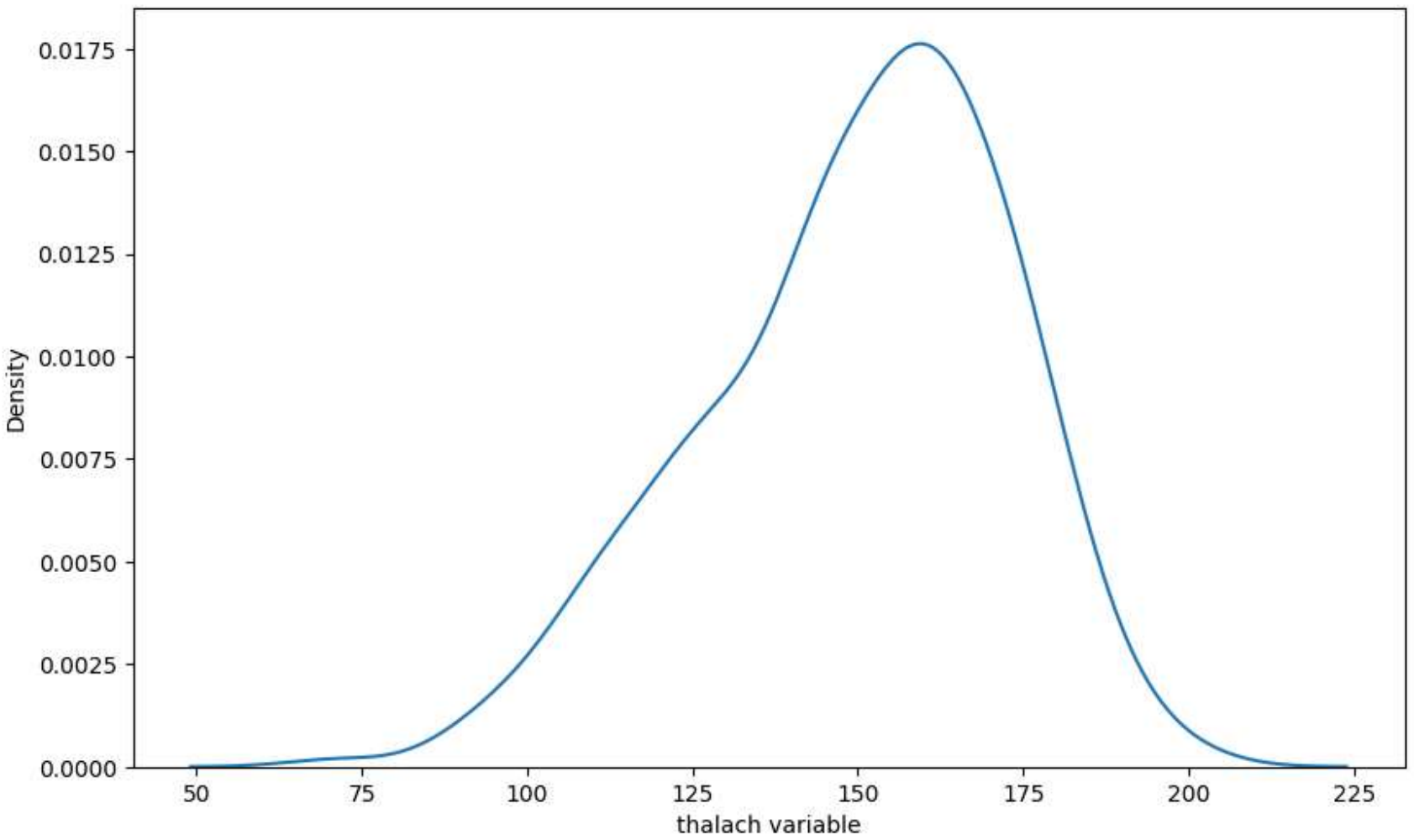


Seaborn Kernel Density Estimation(KDE) plot

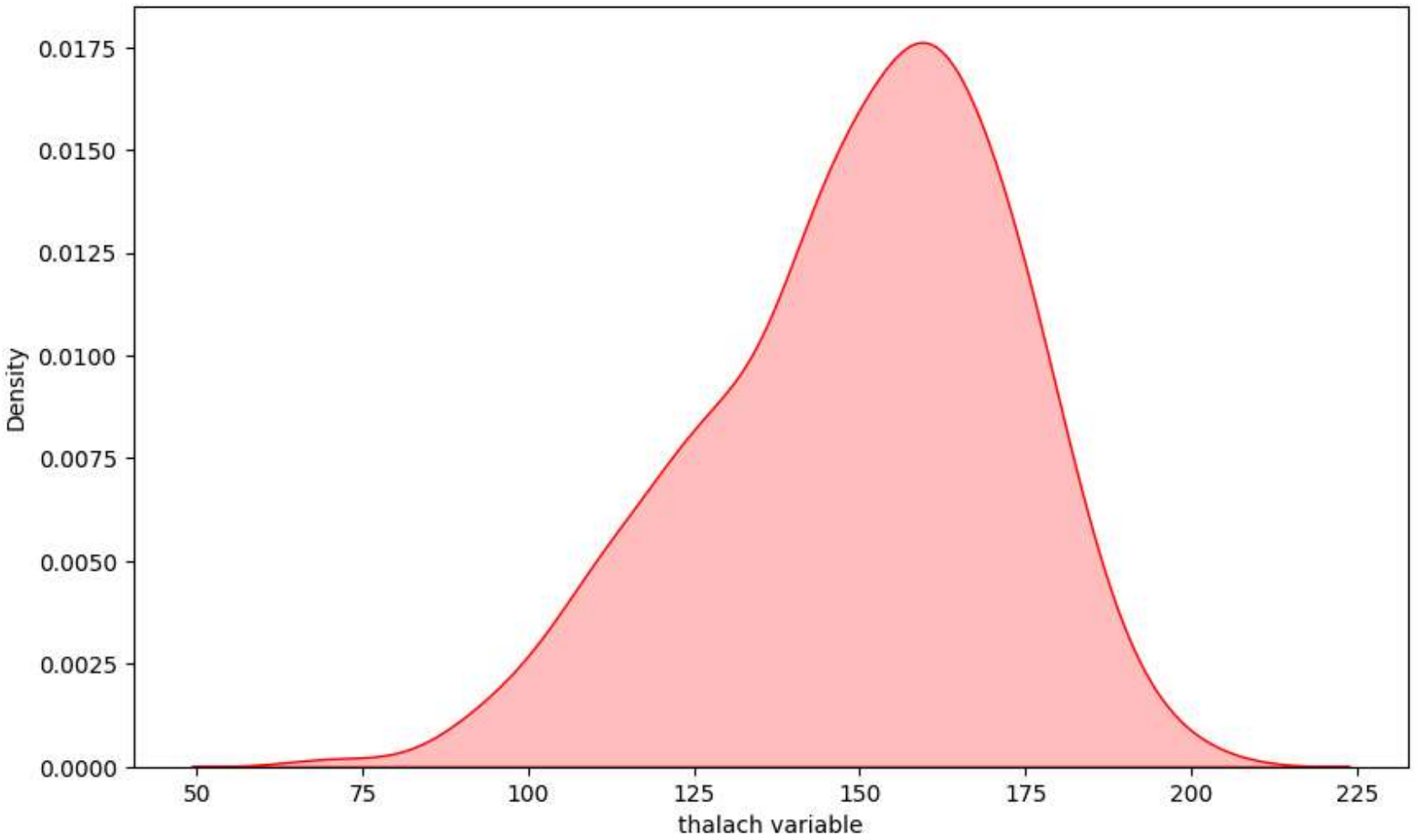
- the kernel density estimate (kde) plot is useful tool for plotting the shape of a distribution

In [36]:
```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
x = pd.Series(x, name="thalach variable")
```

```
ax = sns.kdeplot(x)
plt.show()
```



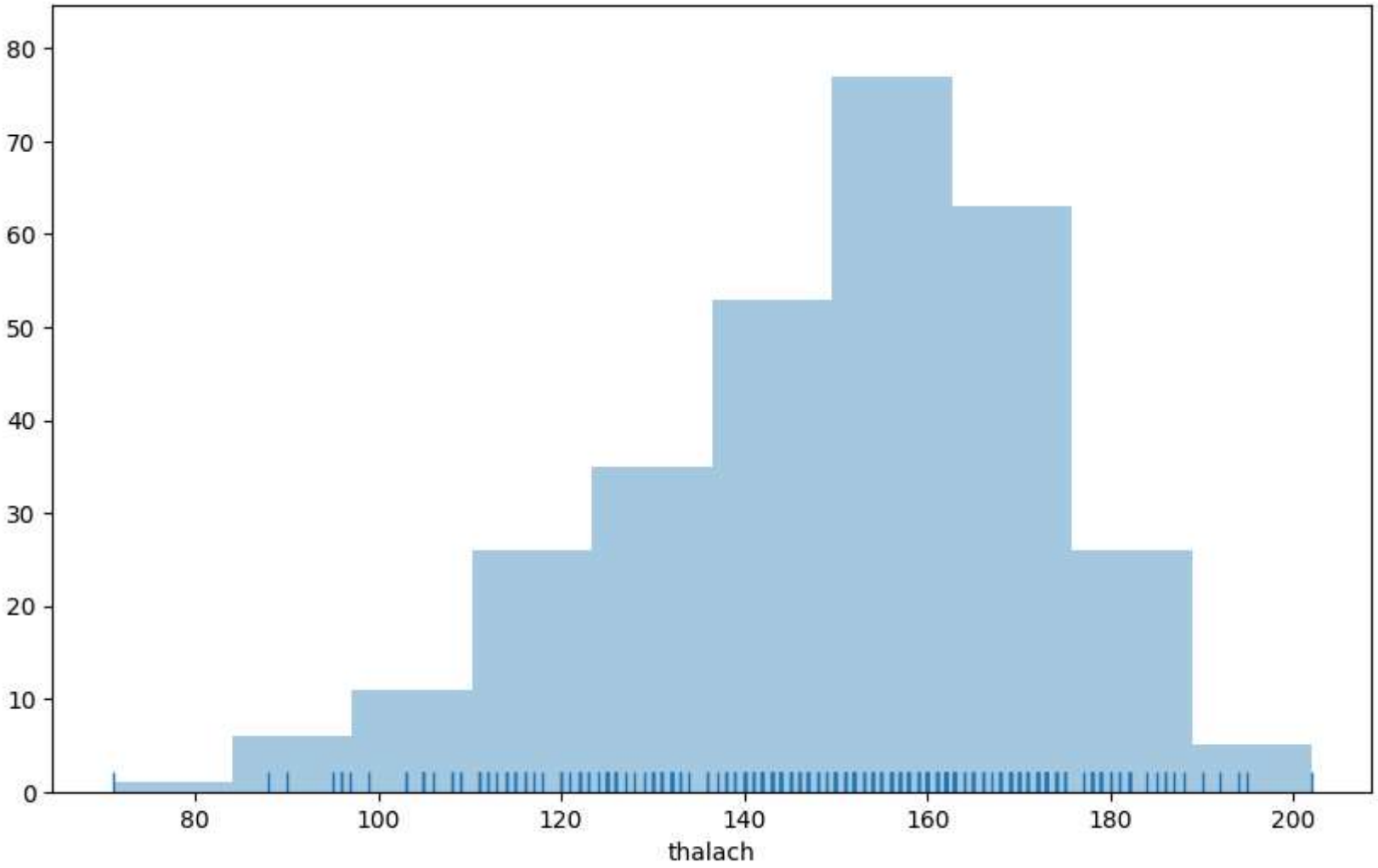
In [37]:
```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
x = pd.Series(x, name="thalach variable")
ax = sns.kdeplot(x, shade = True , color = 'r')
plt.show()
```



Histogram

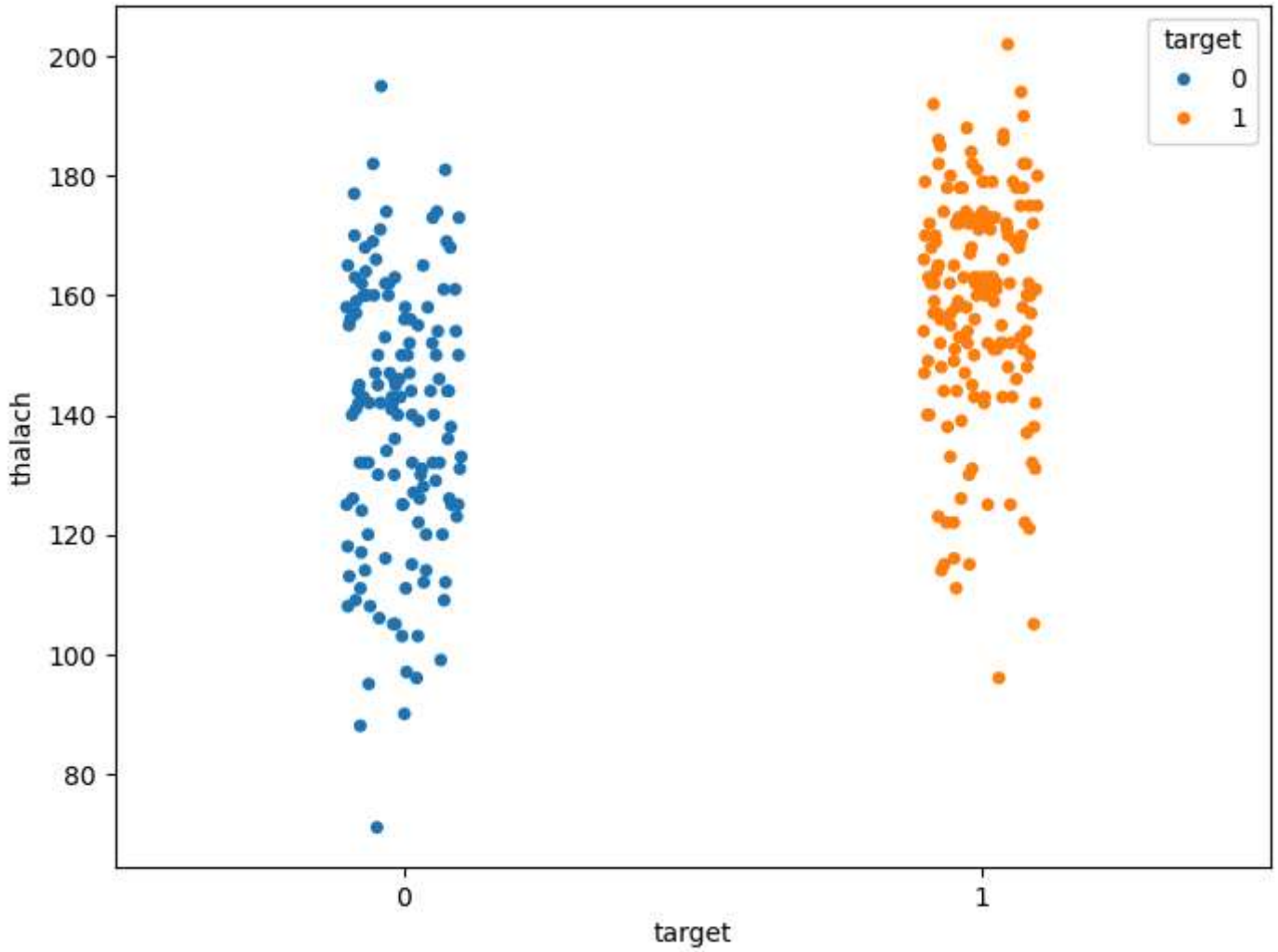
- represents the distributions of the data and shows the number of observations that fall in each bin

In [38]:
```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
ax = sns.distplot(x, kde = False, rug = True ,bins=10)
plt.show()
```

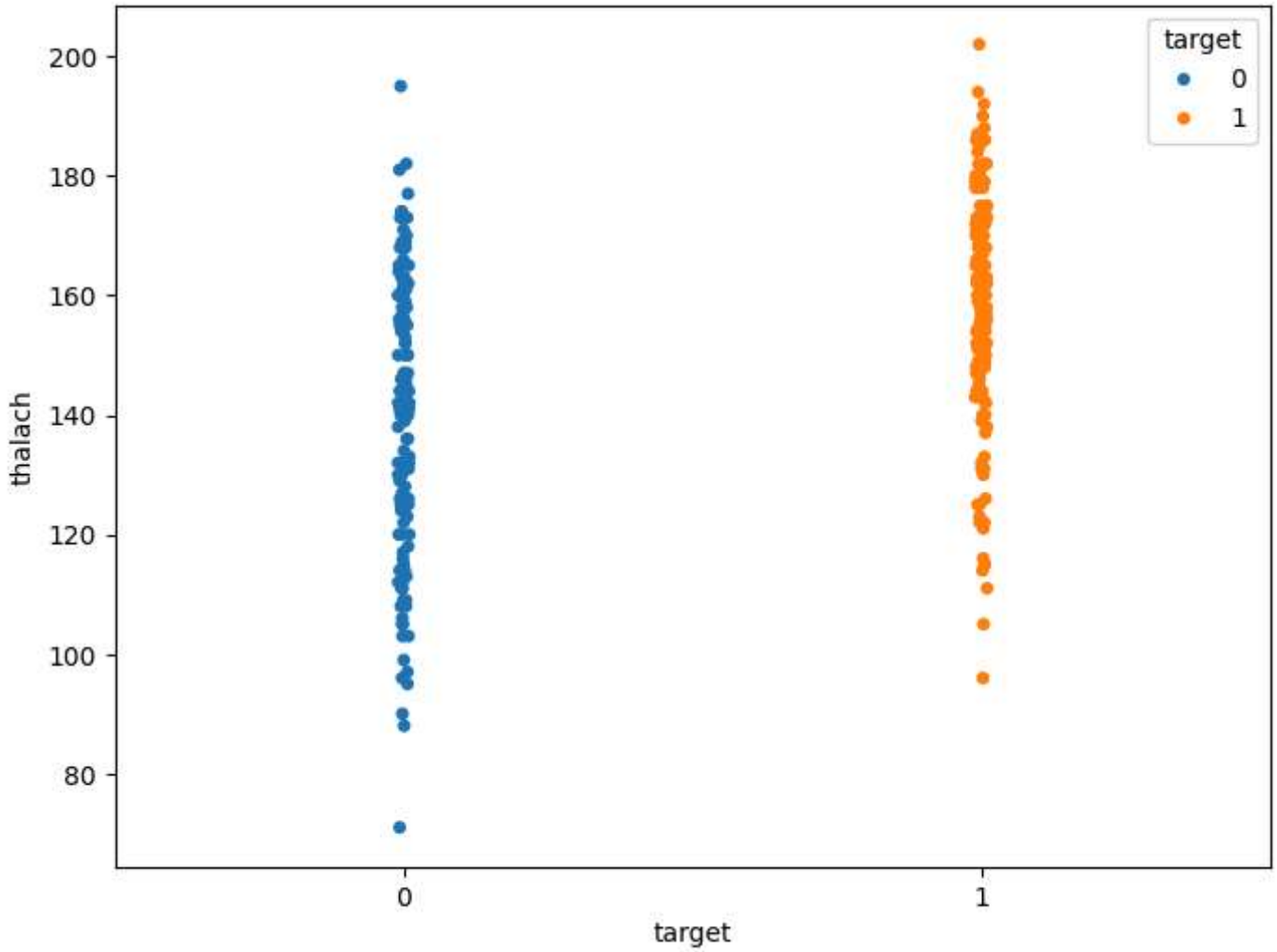


Visualize frequency distribution of `thalach` variable wrt `target`

In [39]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="target", y="thalach", data=df,hue = 'target')
plt.show()
```

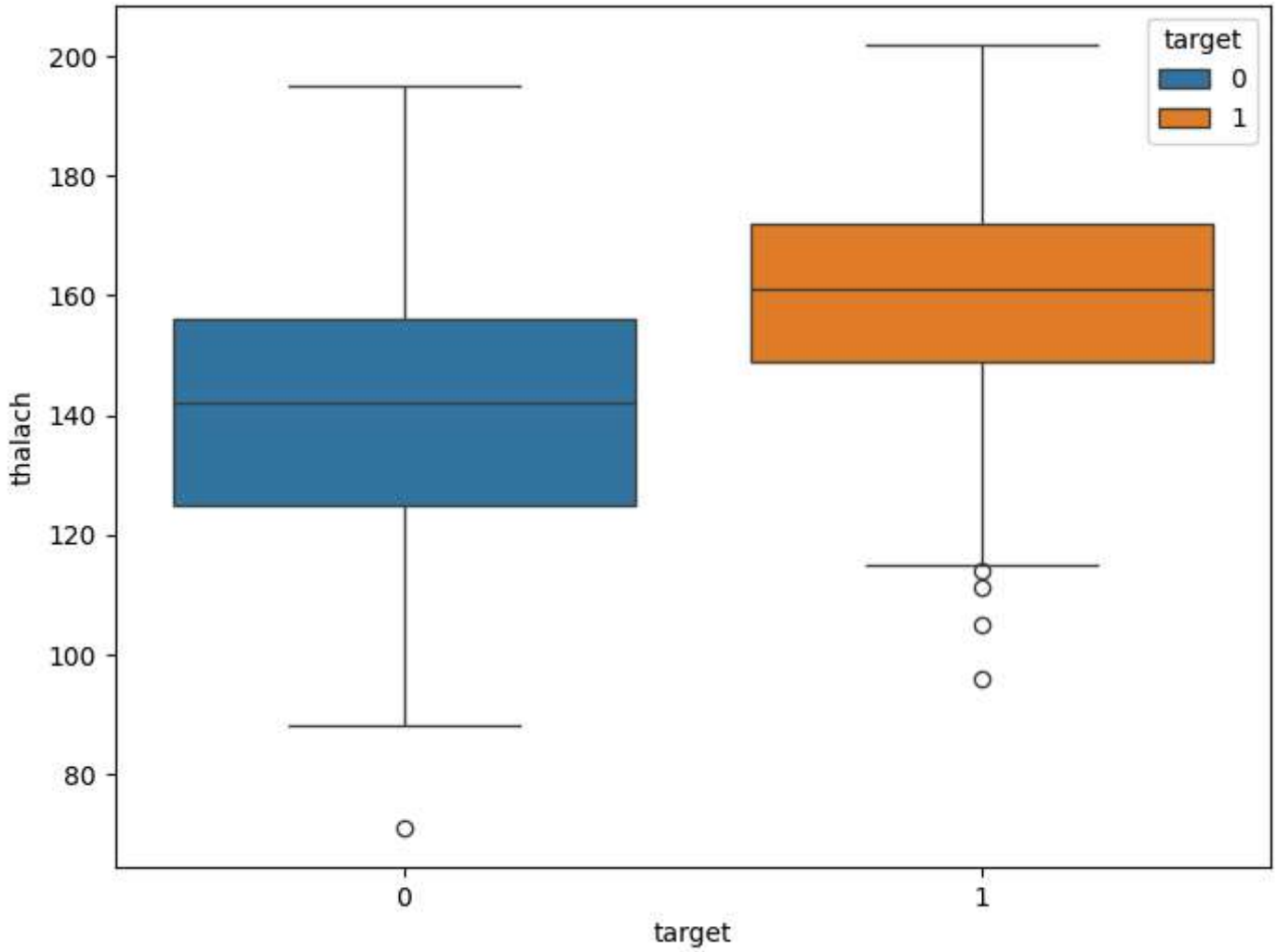


In [40]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="target", y="thalach", data=df , jitter =0.01,hue = 'target')
plt.show()
```



Visualize distribution of `thalach` variable wrt `target` with boxplot

In [41]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x="target", y="thalach", data=df , hue='target')
plt.show()
```



comment

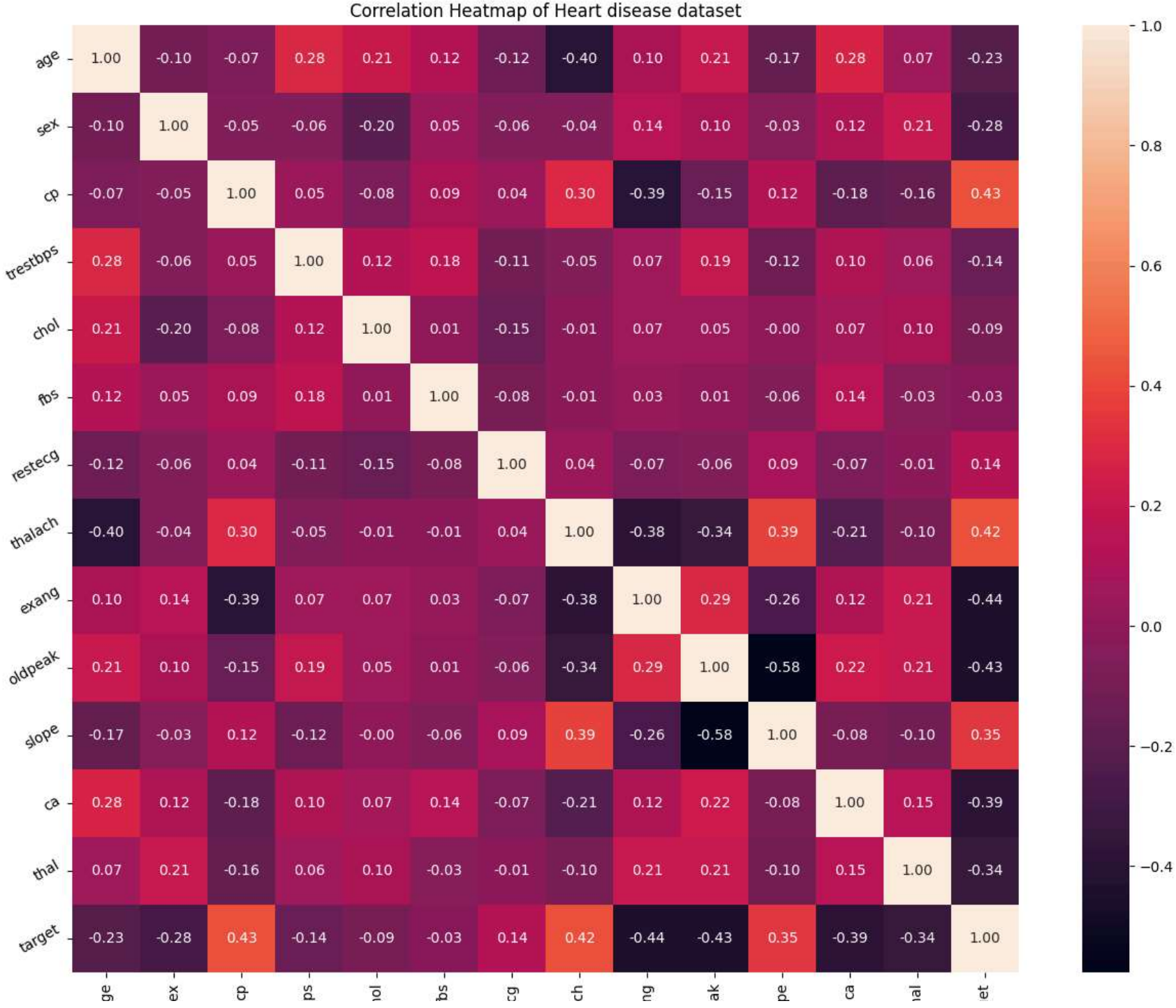
- the cp and thalach variables are mildly positively correlated with target variable
- we can see that the talach variable is slightly negatively skewed

Multivariate analysis

- multivariate analysis is to discover patterns and relationships in the dataset
- I will use heatmap and pair plot to discover the patterns and relationships in the dataset

Heat Map

In [42]:
```
plt.figure(figsize=(16,12))
plt.title('Correlation Heatmap of Heart 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()
```


Correlation Heatmap of Heart disease dataset
age 1.00 -0.10 -0.07 0.28 0.21 0.12 -0.12 -0.40 0.10 0.21 -0.17 0.28 0.07 -0.23
sex -0.10 1.00 -0.05 -0.06 -0.20 0.05 -0.06 -0.04 0.14 0.10 -0.03 0.12 0.21 -0.28
cp -0.07 -0.05 1.00 0.05 -0.08 0.09 0.04 0.30 -0.39 -0.15 0.12 -0.18 -0.16 0.43
trestbps 0.28 -0.06 0.05 1.00 0.12 0.18 -0.11 -0.05 0.07 0.19 -0.12 0.10 0.06 -0.14
chol 0.21 -0.20 -0.08 0.12 1.00 0.01 -0.15 -0.01 0.07 0.05 -0.00 0.07 0.10 -0.09
fbs 0.12 0.05 0.09 0.18 0.01 1.00 -0.08 -0.01 0.03 0.01 -0.06 0.14 -0.03 -0.03
restecg -0.12 -0.06 0.04 -0.11 -0.15 -0.08 1.00 0.04 -0.07 -0.06 0.09 -0.07 -0.01 0.14
thalach -0.40 -0.04 0.30 -0.05 -0.01 -0.01 0.04 1.00 -0.38 -0.34 0.39 -0.21 -0.10 0.42
exang 0.10 0.14 -0.39 0.07 0.07 0.03 -0.07 -0.38 1.00 0.29 -0.26 0.12 0.21 -0.44
oldpeak 0.21 0.10 -0.15 0.19 0.05 0.01 -0.06 -0.34 0.29 1.00 -0.58 0.22 0.21 -0.43
slope -0.17 -0.03 0.12 -0.12 -0.00 -0.06 0.09 0.39 -0.26 -0.58 1.00 -0.08 -0.10 0.35
ca 0.28 0.12 -0.18 0.10 0.07 0.14 -0.07 -0.21 0.12 0.22 -0.08 1.00 0.15 -0.39
thal 0.07 0.21 -0.16 0.06 0.10 -0.03 -0.01 -0.10 0.21 0.21 -0.10 0.15 1.00 -0.34
target -0.23 -0.28 0.43 -0.14 -0.09 -0.03 0.14 0.42 -0.44 -0.43 0.35 -0.39 -0.34 1.00
ge ex cp ps ol bs cg ch ng ak pe ca al et
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

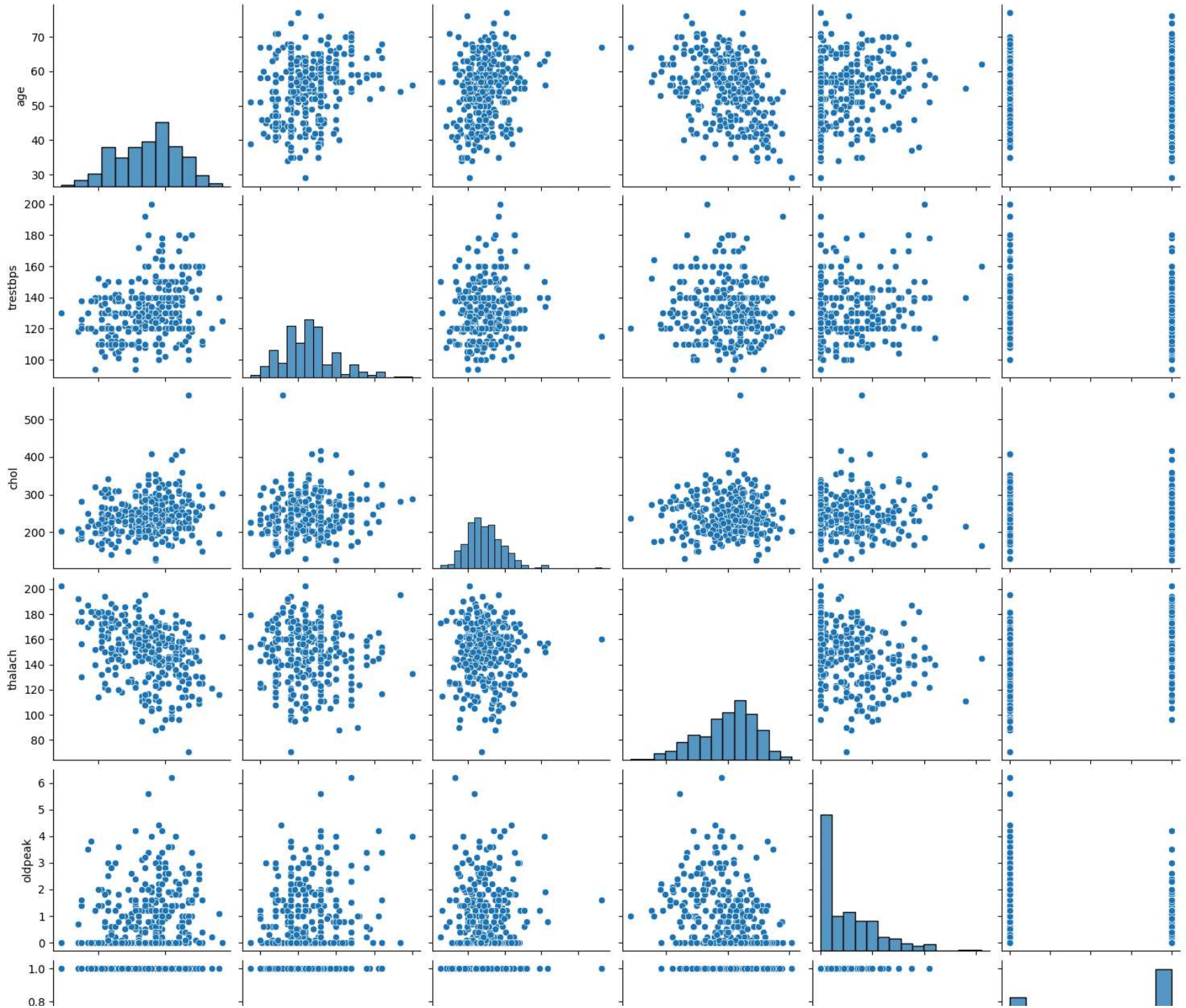
Interpretation

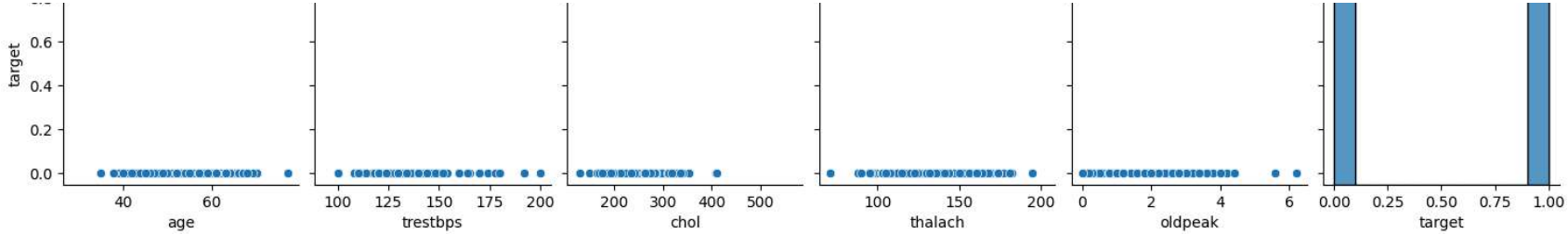
From the above correlation heat map, we can conclude that :-

- `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 [43]:
```
num_var = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'target' ]
sns.pairplot(df[num_var], kind='scatter',diag_kind='hist')
plt.show()
```


age
trestbps
chol
thalach
oldpeak



Comment

- 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 [44]:
```
df['age'].nunique()
```

Out[44]:
```
41
```

view statistical summary of age variable

In [45]:
```
df['age'].describe()
```

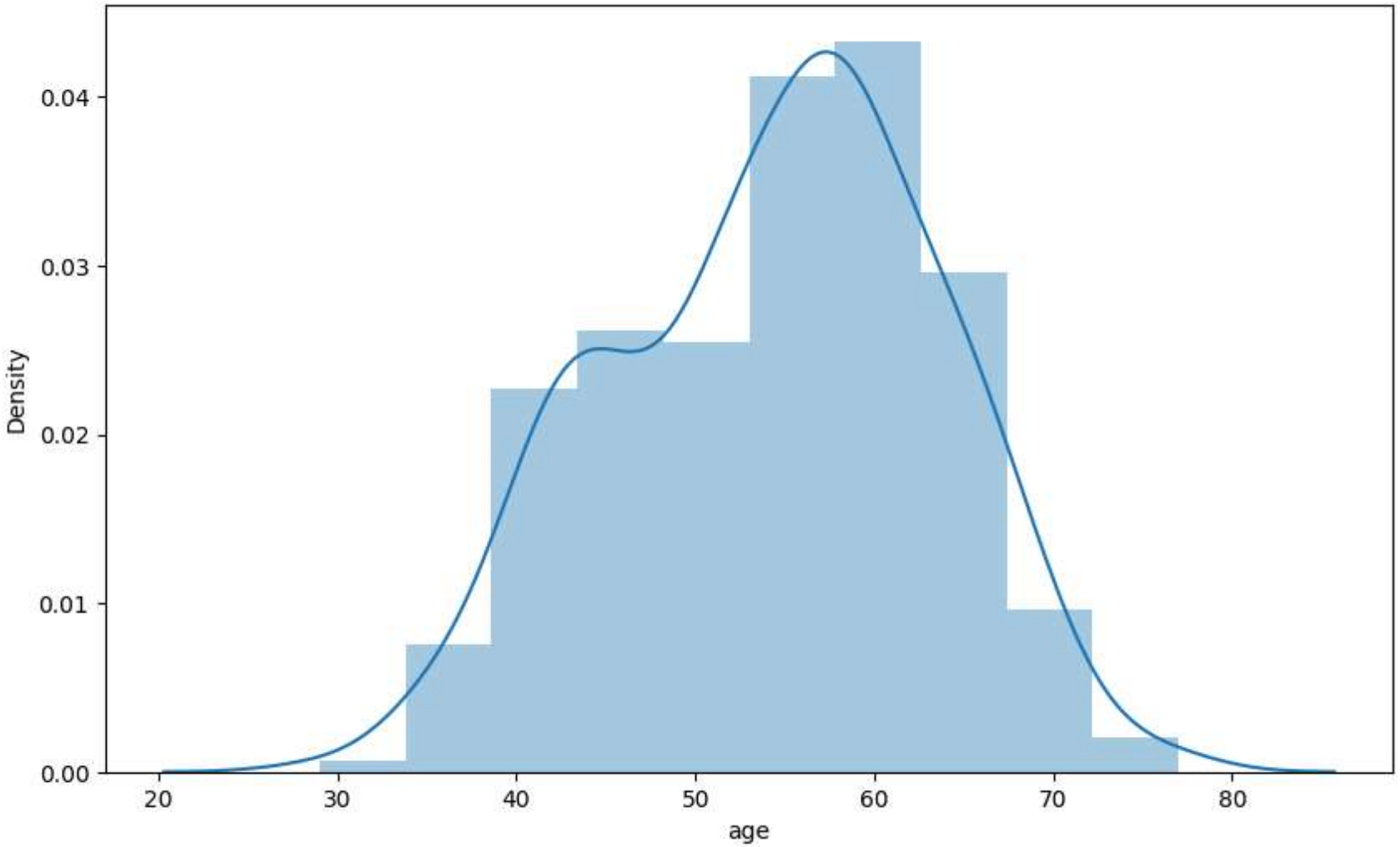
Out[45]:
```
count    303.000000
mean      54.366337
std        9.082101
min       29.000000
25%       47.500000
50%       55.000000
75%       61.000000
max       77.000000
Name: age, dtype: float64
```

Interpretation

- 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 [46]:
```
f, ax = plt.subplots(figsize=(10,6))
x = df['age']
ax = sns.distplot(x, bins=10)
plt.show()
```

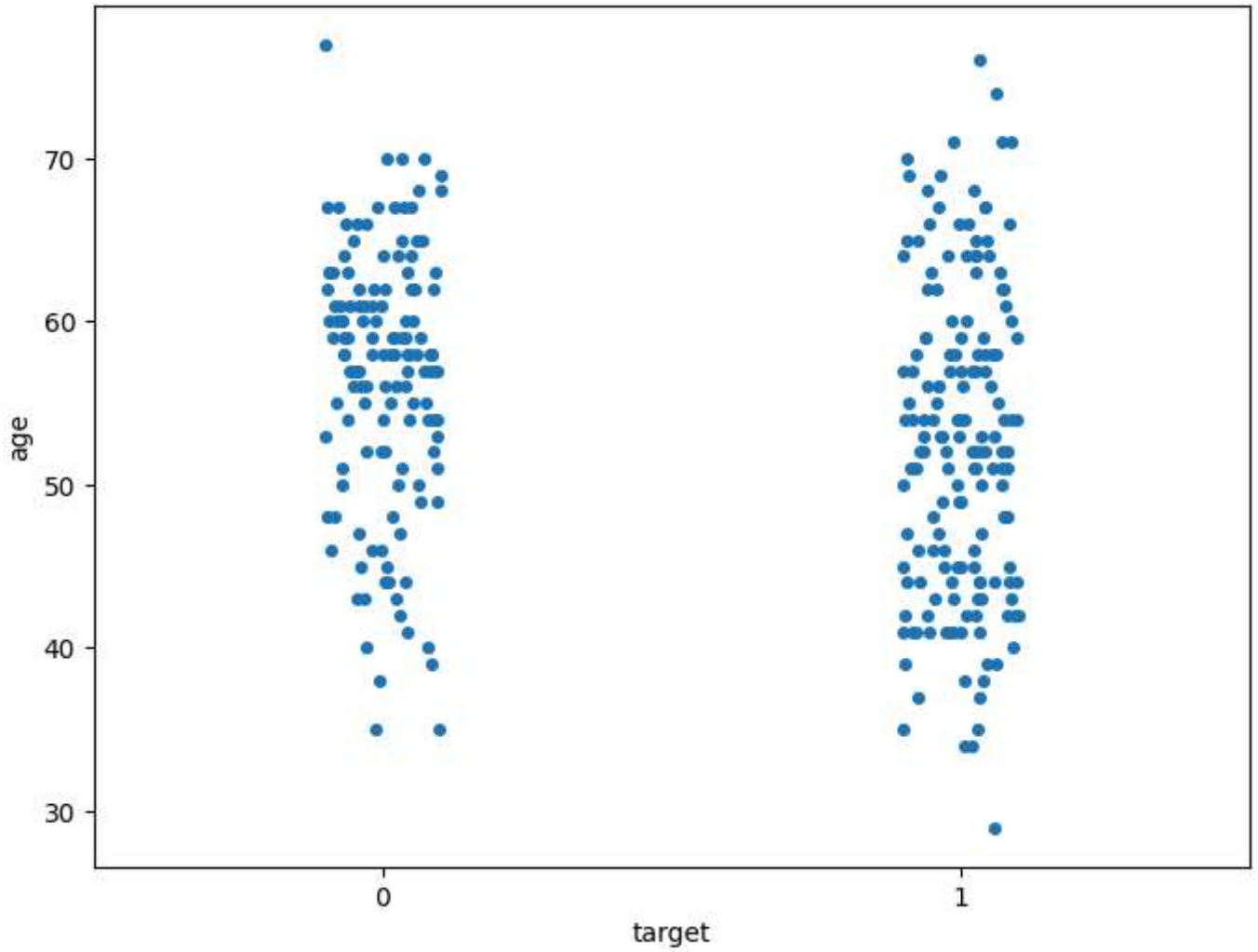


interpretation

- The age variable distribution is approx normal

Analyze age and target variable

In [47]:
```
f ,ax = plt.subplots(figsize=(8,6))
sns.stripplot(x='target',y='age',data=df)
plt.show()
```

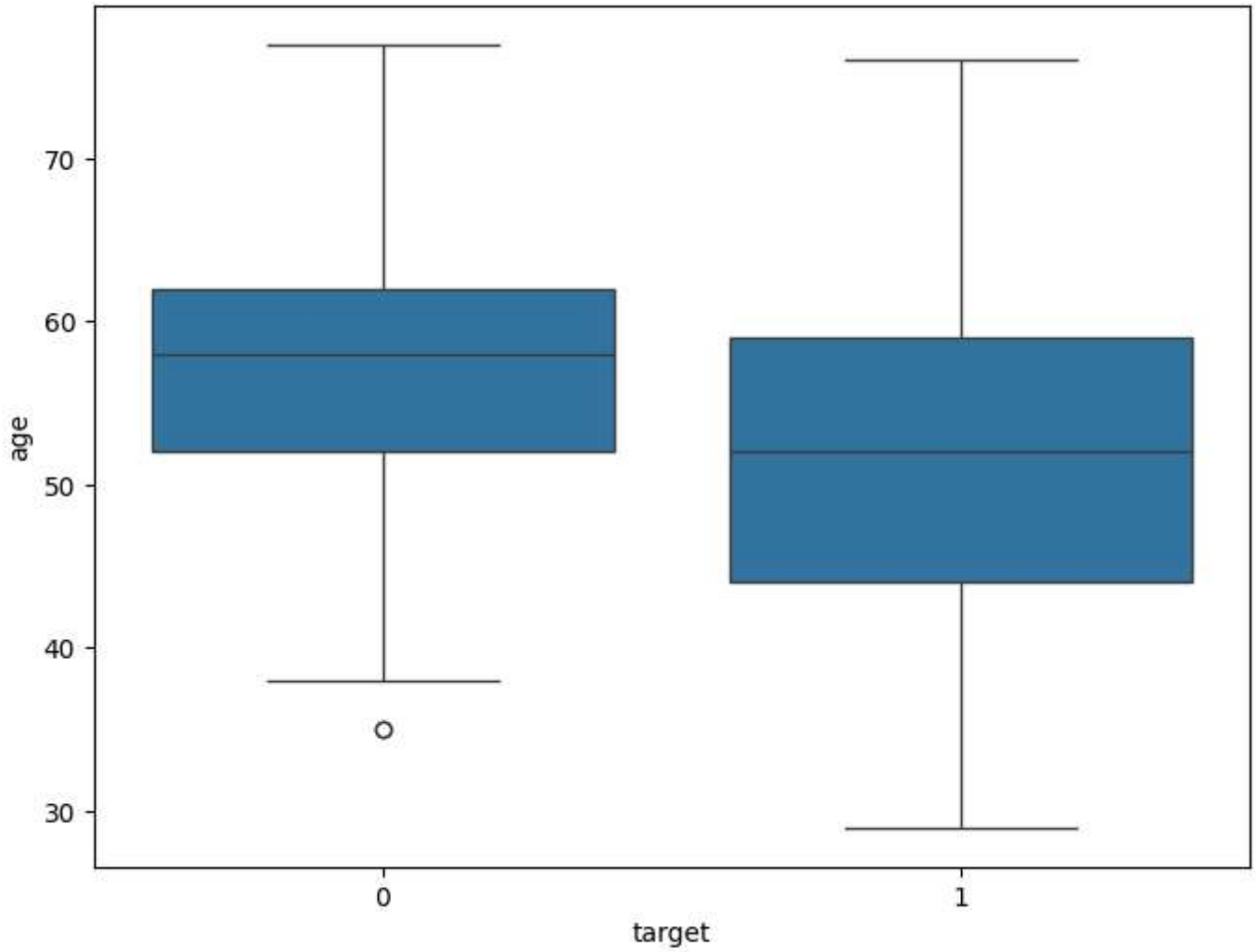


Interpretation

- 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.

Visualize distribution of age variable wrt target with boxplot

In [48]:
```
f ,ax = plt.subplots(figsize=(8,6))
sns.boxplot(x='target',y='age',data=df)
plt.show()
```

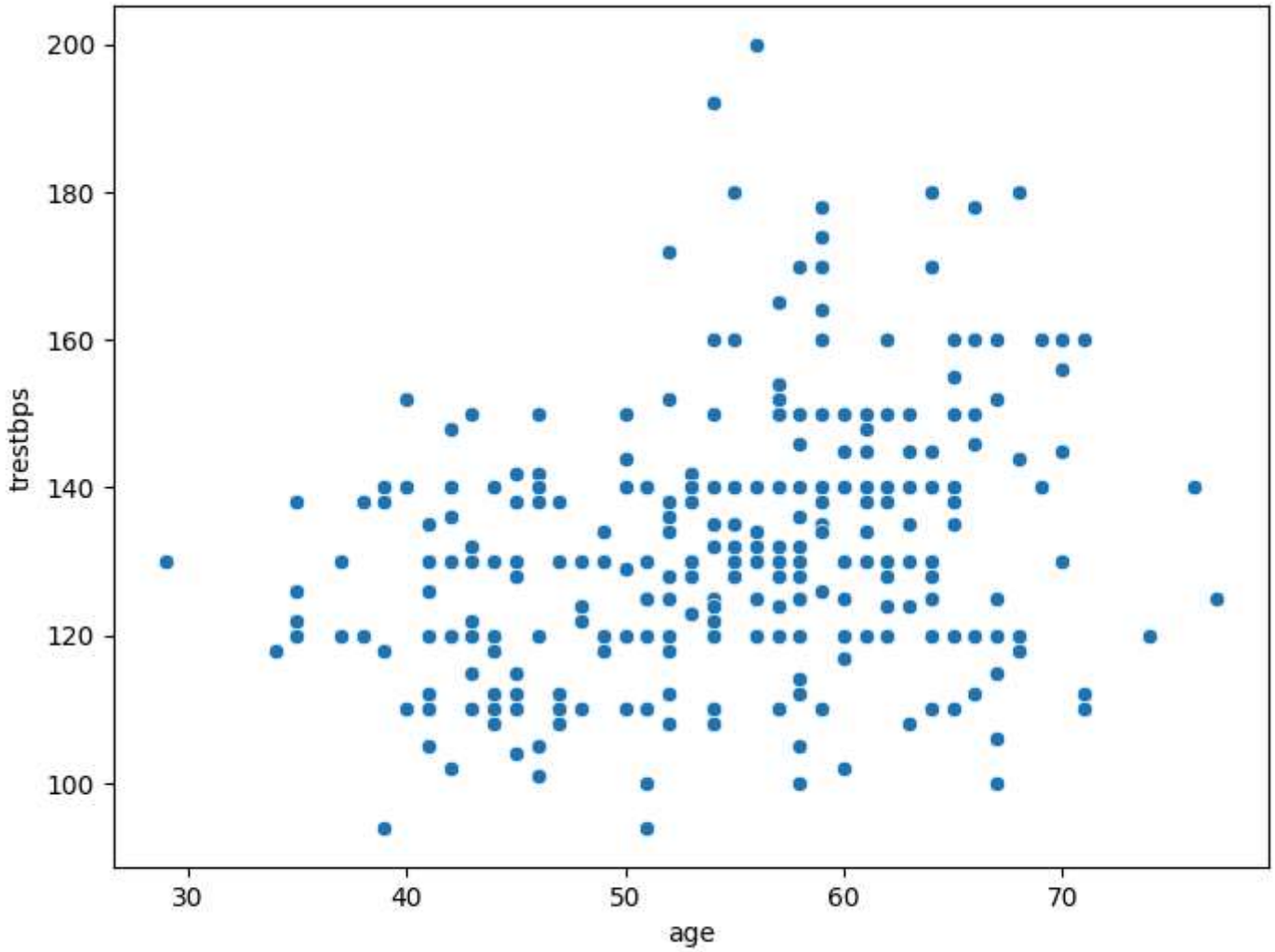


Interpretation

- 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

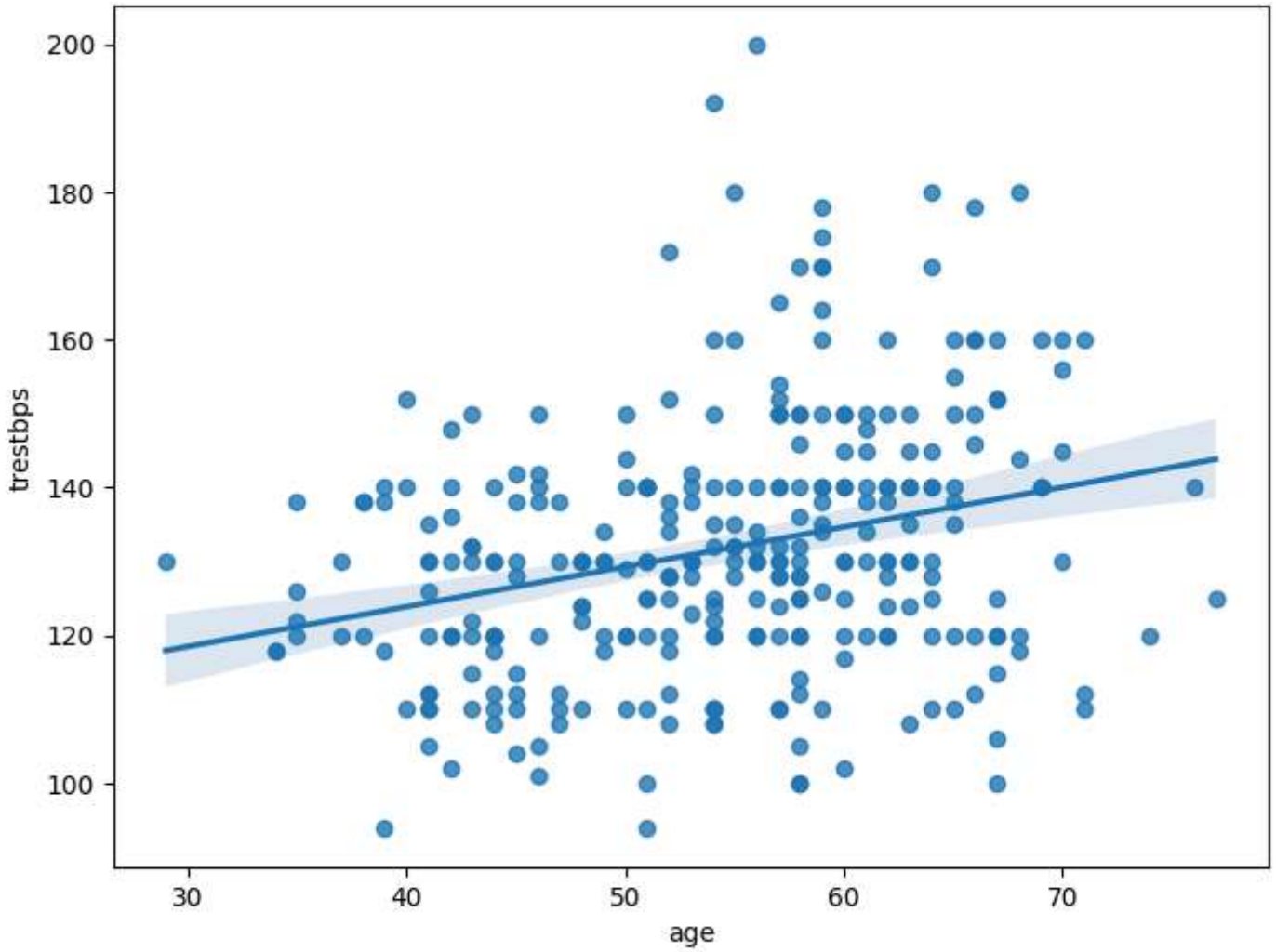
In [49]:
```
f , ax = plt.subplots(figsize = (8,6))
ax = sns.scatterplot(x='age',y = 'trestbps',data=df)
plt.show()
```



Interpretation

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

In [50]:
```
f , ax = plt.subplots(figsize = (8,6))
ax = sns.regplot(x='age',y = 'trestbps',data=df)
plt.show()
```



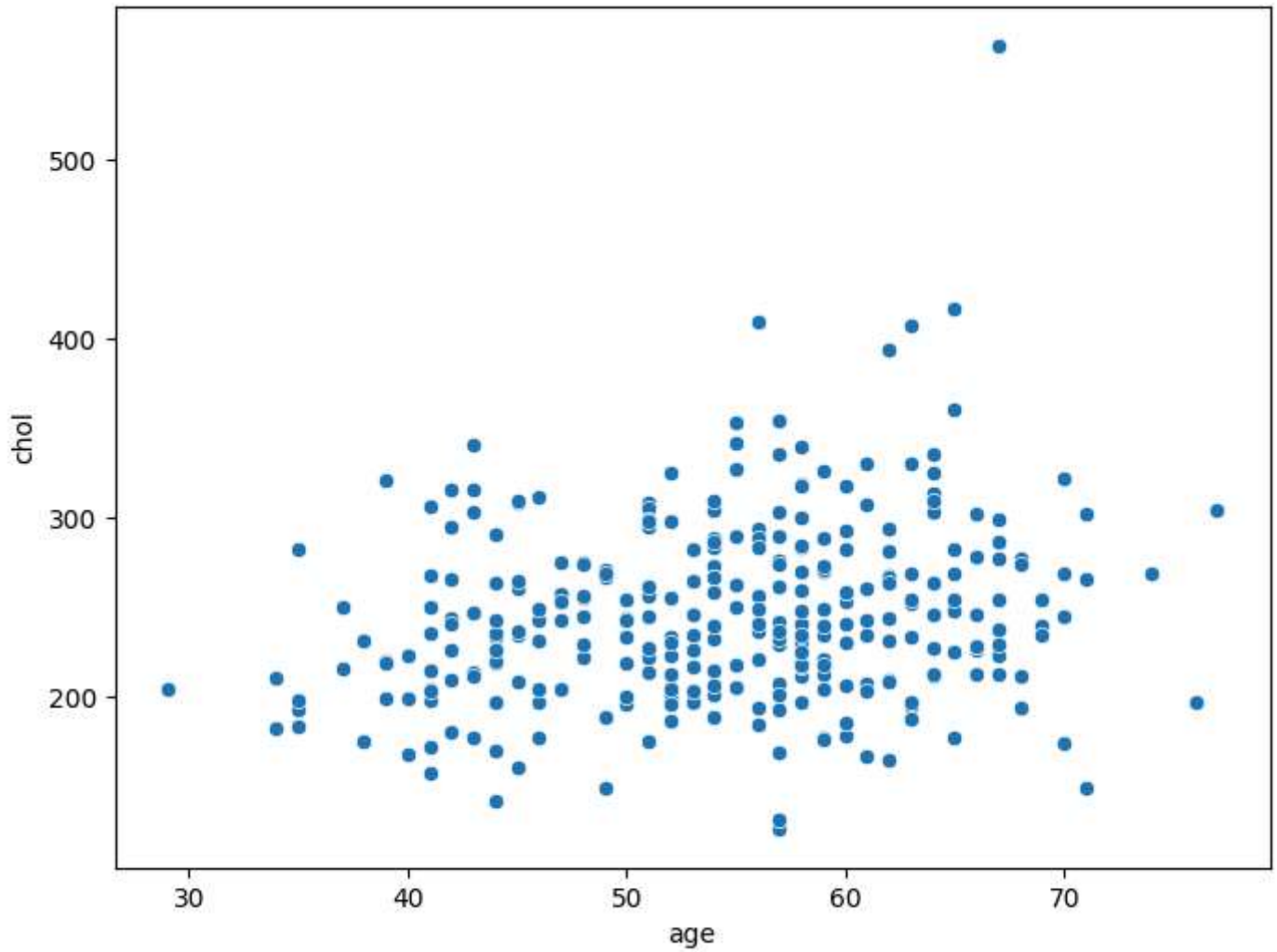
Interpretation

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

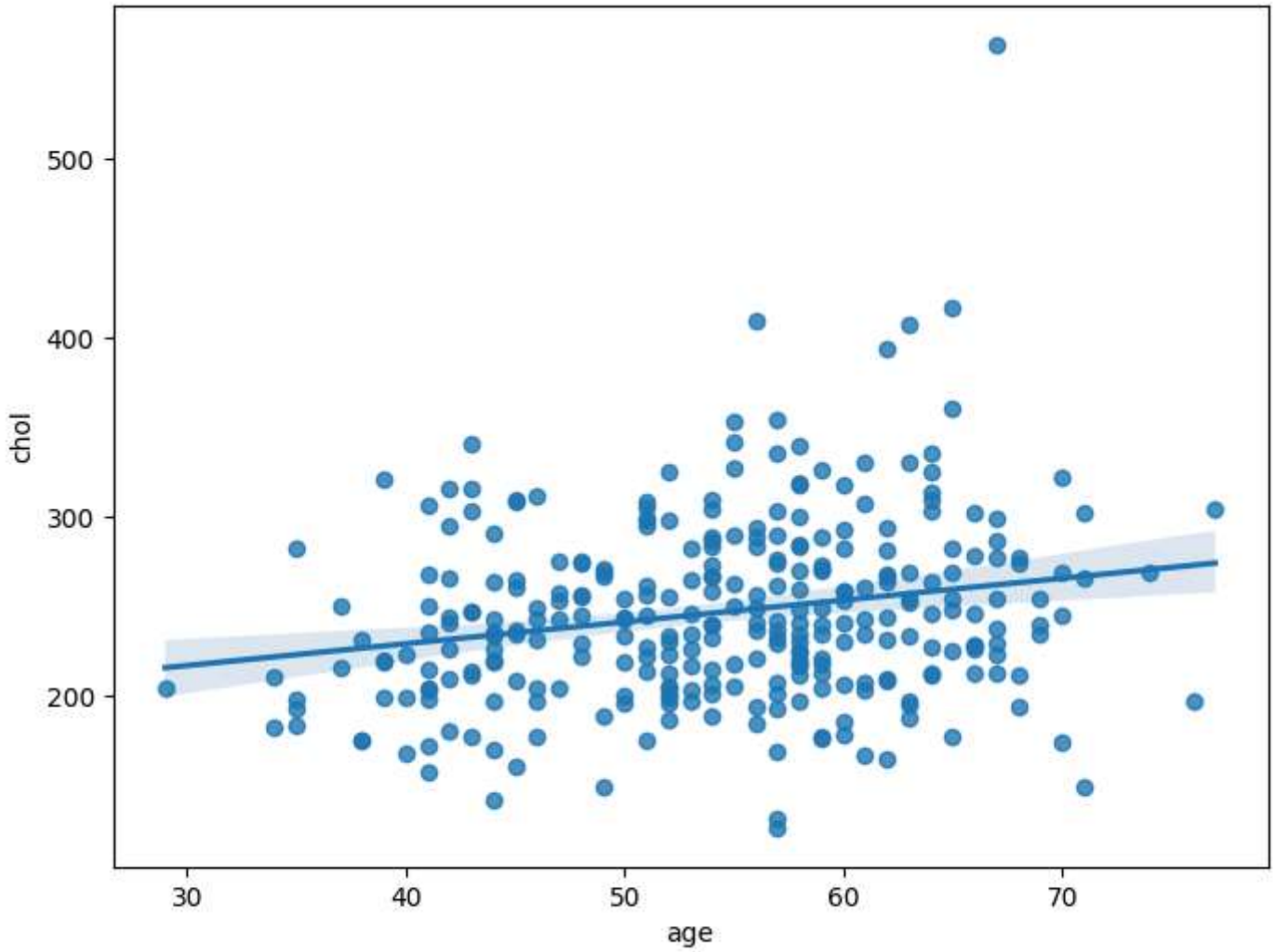
Analyze age and chol variable

In [51]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.scatterplot(x="age", y="chol", data=df)
```

```
plt.show()
```



In [52]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="age", y="chol", data=df)
plt.show()
```



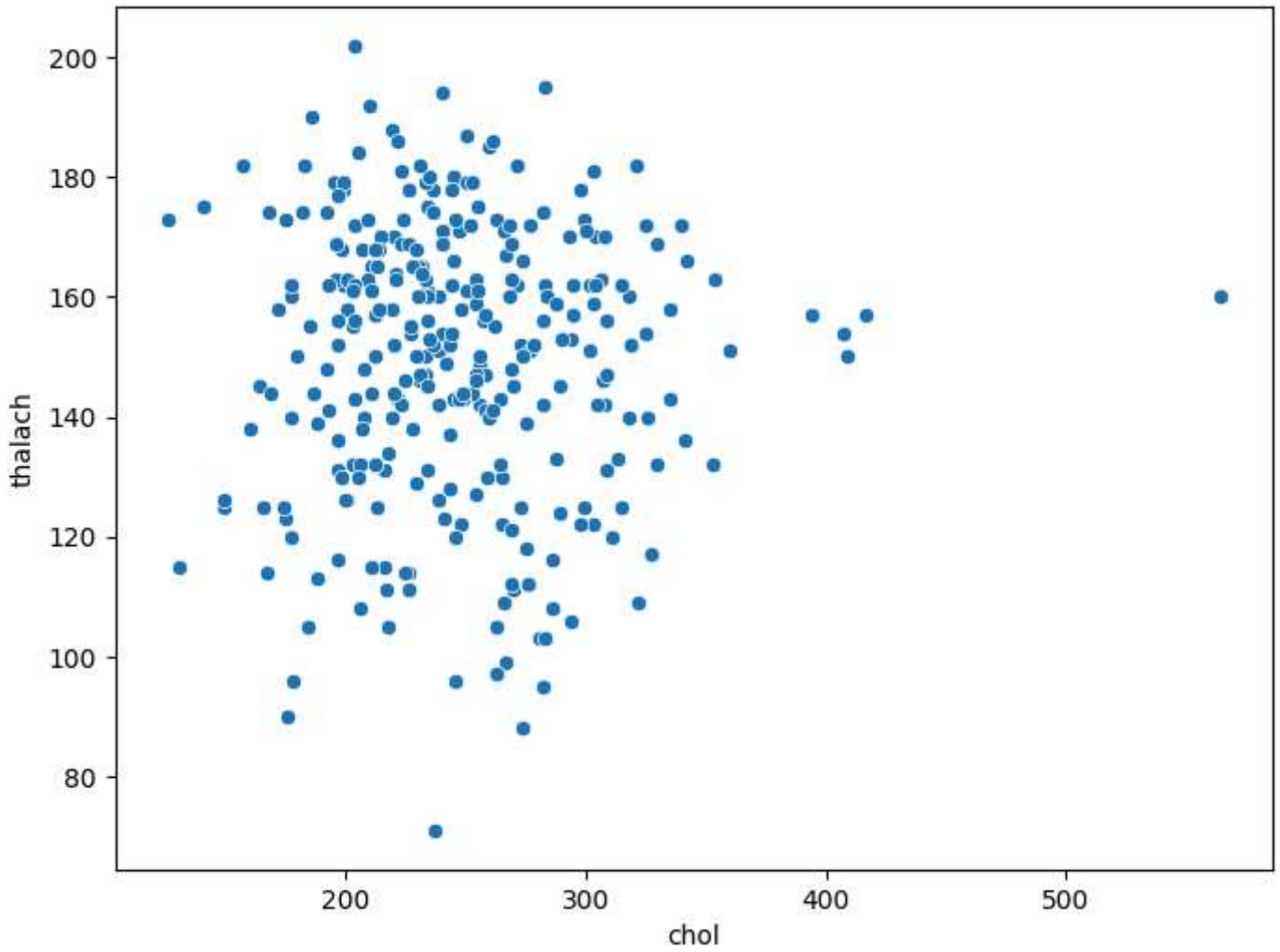
Interpretation

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

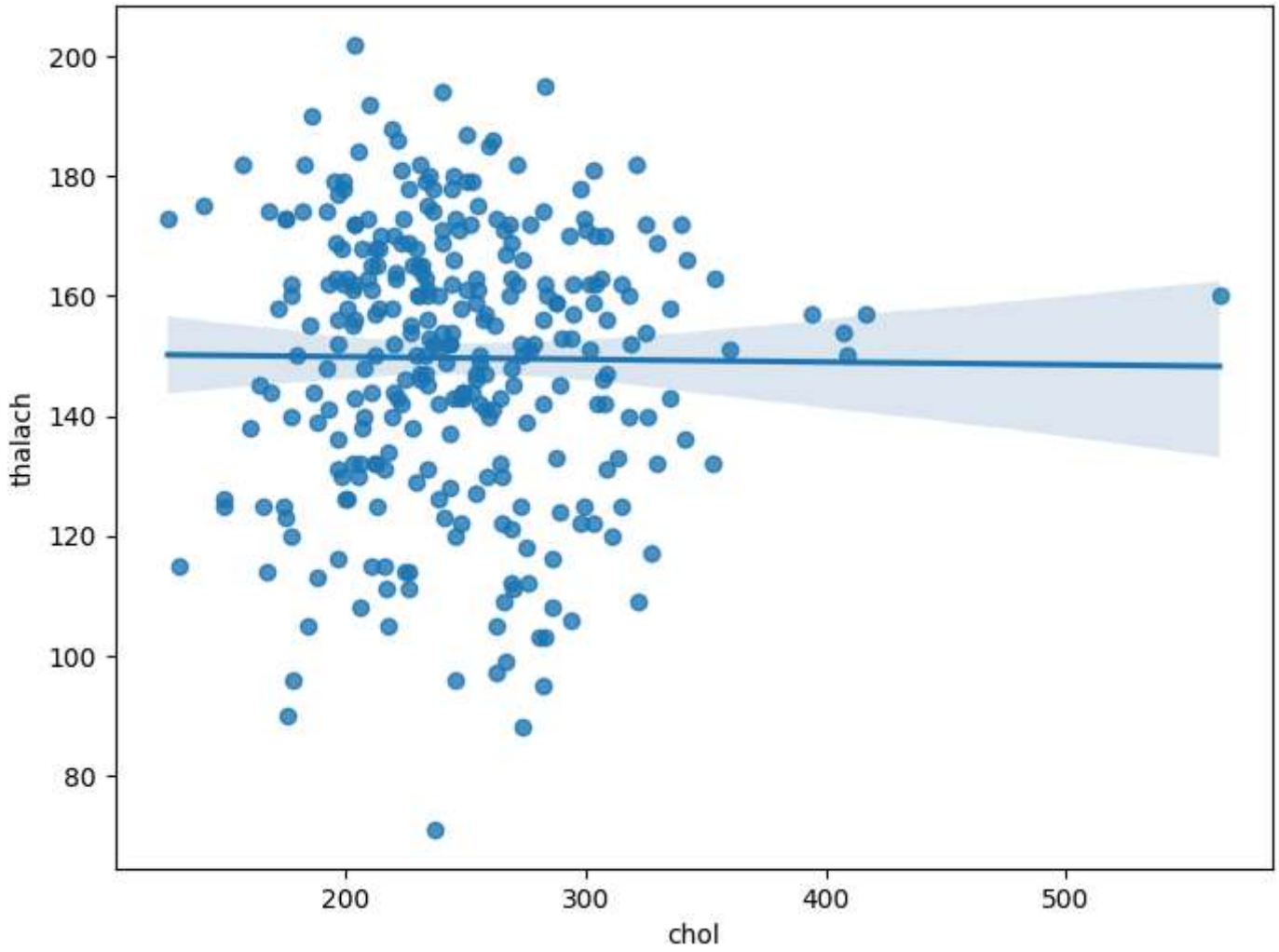
Analyze chol and thalach variable

In [53]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.scatterplot(x="chol", y = "thalach", data=df)
```

```
plt.show()
```



In [54]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="chol", y="thalach", data=df)
plt.show()
```



Interpretation

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

Dealing with missing values

check for missing values

In [55]:
```
df.isnull().sum()
```

Out[55]:
```
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
```

Interpretation

We can see that there are no missing values in the dataset.

Check with ASSERT statement

- **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 [56]:
```
#assert that there are no missing values in the dataframe

assert pd.notnull(df).all().all()
```

In [57]:
```
#assert all values are greater than or equal to 0

assert (df >= 0).all().all()
```

Interpretation

- 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.

Outlier detection

age variable

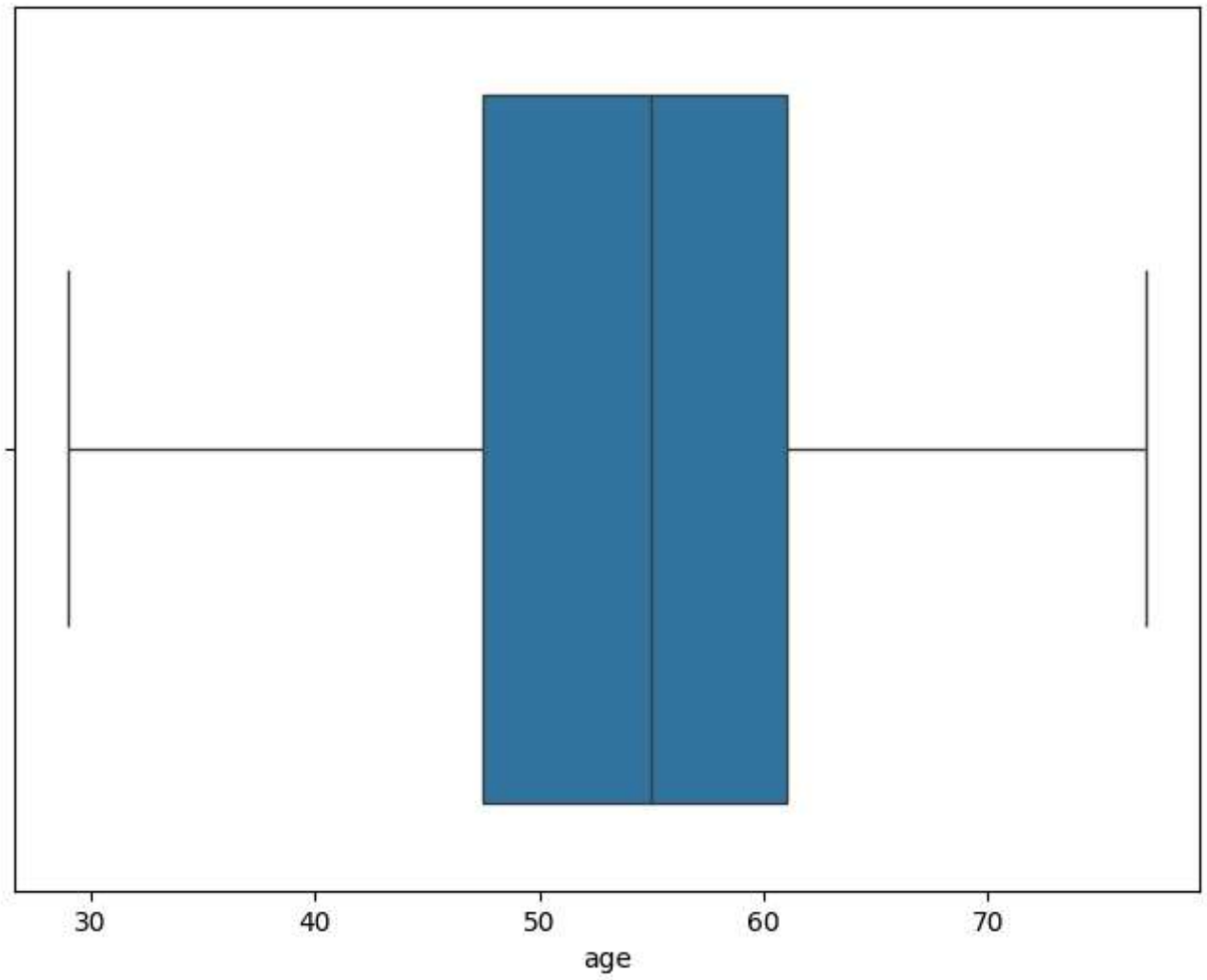
In [58]:
```
df['age'].describe()
```

Out[58]:
```
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
```

box-plot of age variable

In [59]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["age"])
```

```
plt.show()
```



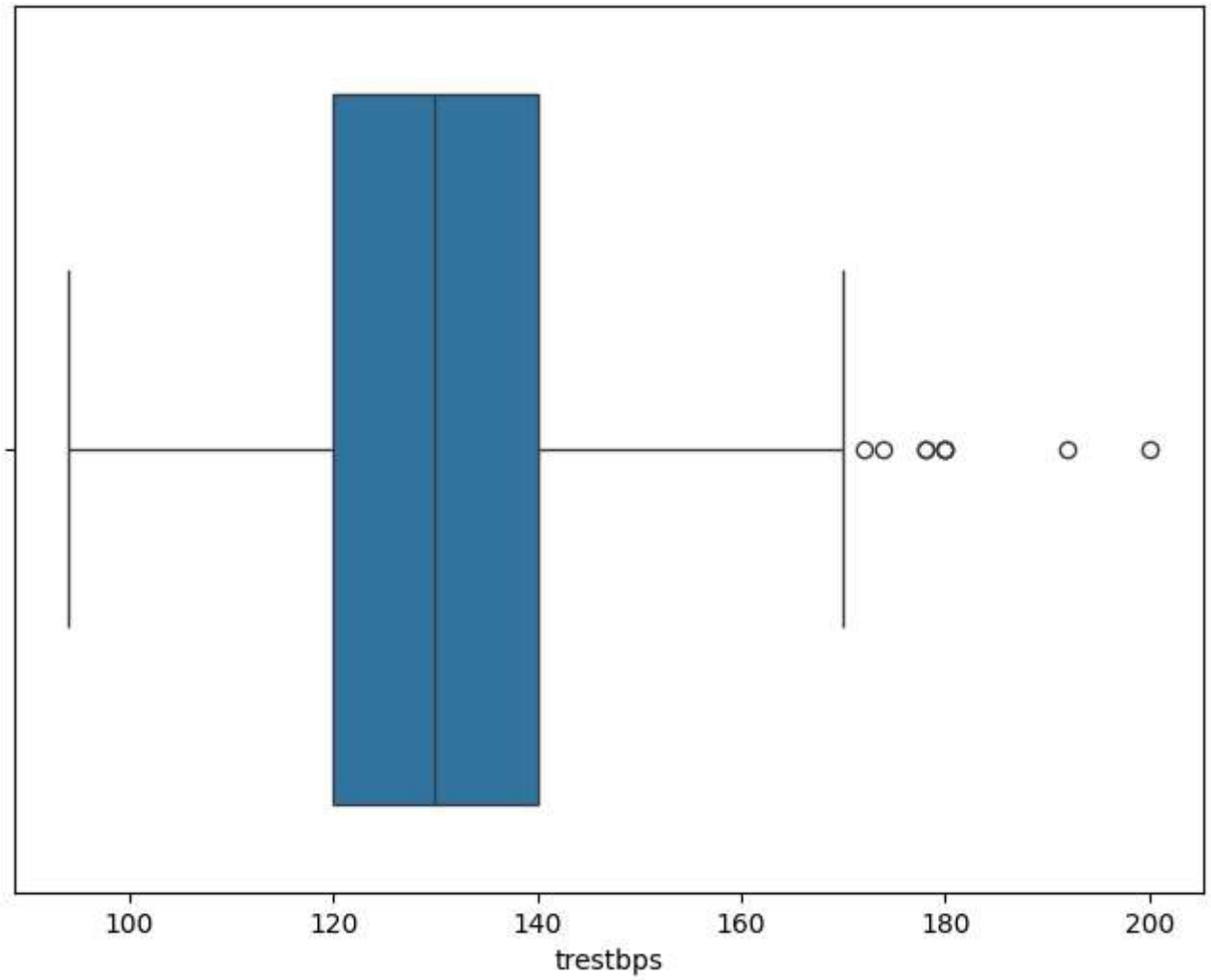
trestbps variable

In [60]:
```
df['trestbps'].describe()
```

Out[60]:
```
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
```

box - plot of trestbps variable

In [61]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["trestbps"])
plt.show()
```



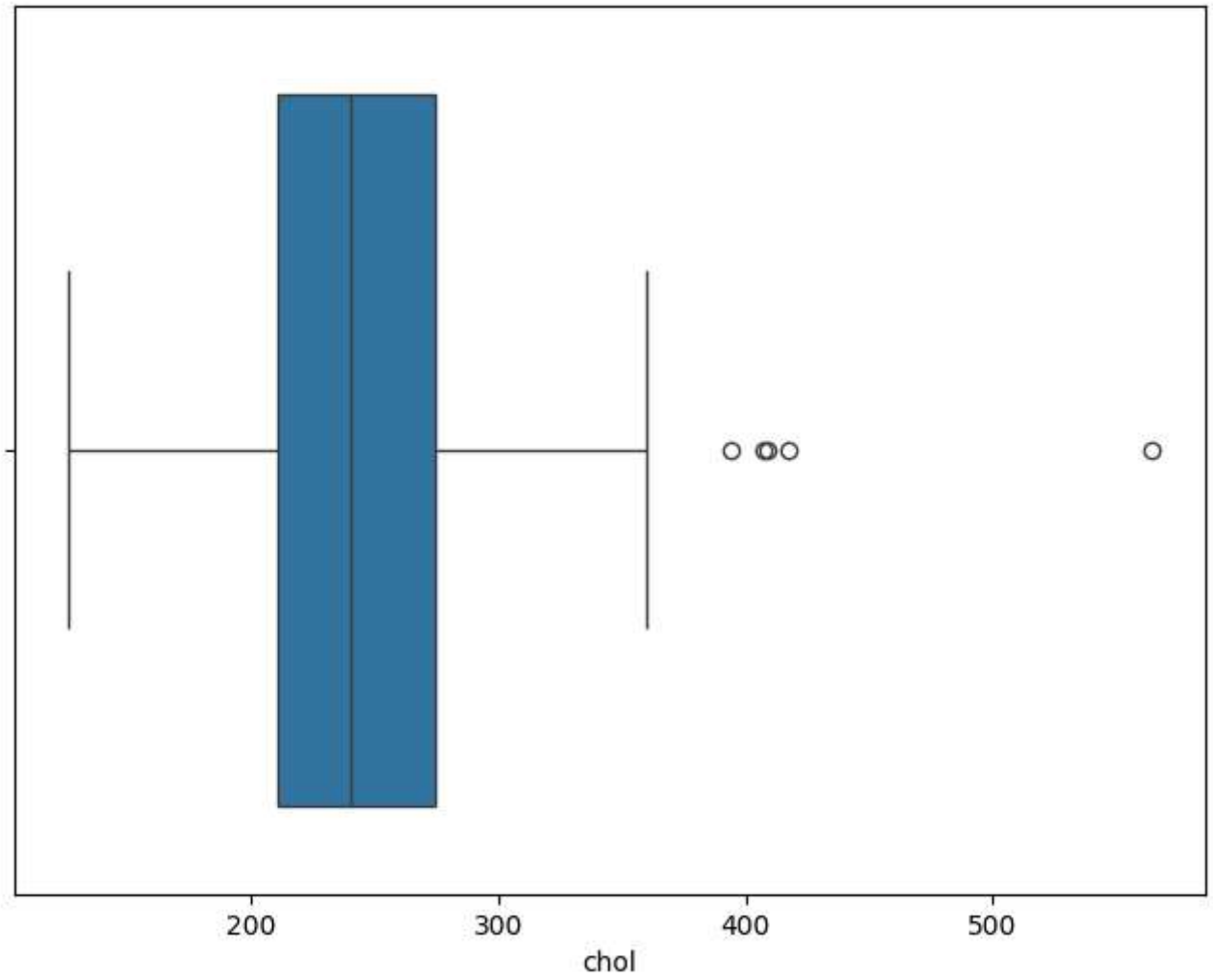
chol variable

In [62]:
```
df['chol'].describe()
```

Out[62]:
```
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
```

box-ploot of chol variable

In [63]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["chol"])
plt.show()
```



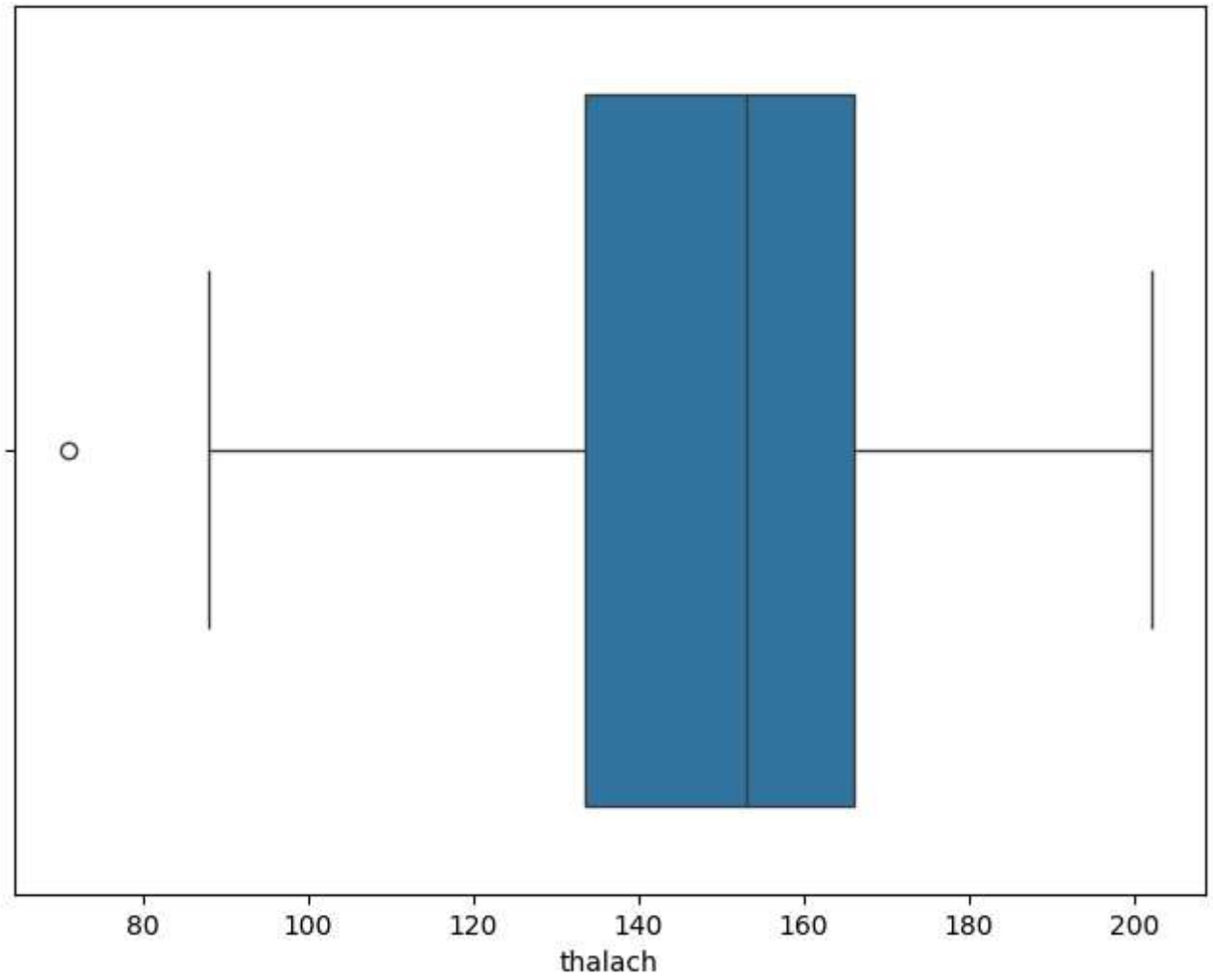
thalach variable

In [64]:
```
df['thalach'].describe()
```

Out[64]:
```
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
```

Box-plot of thalach variable

In [65]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["thalach"])
plt.show()
```



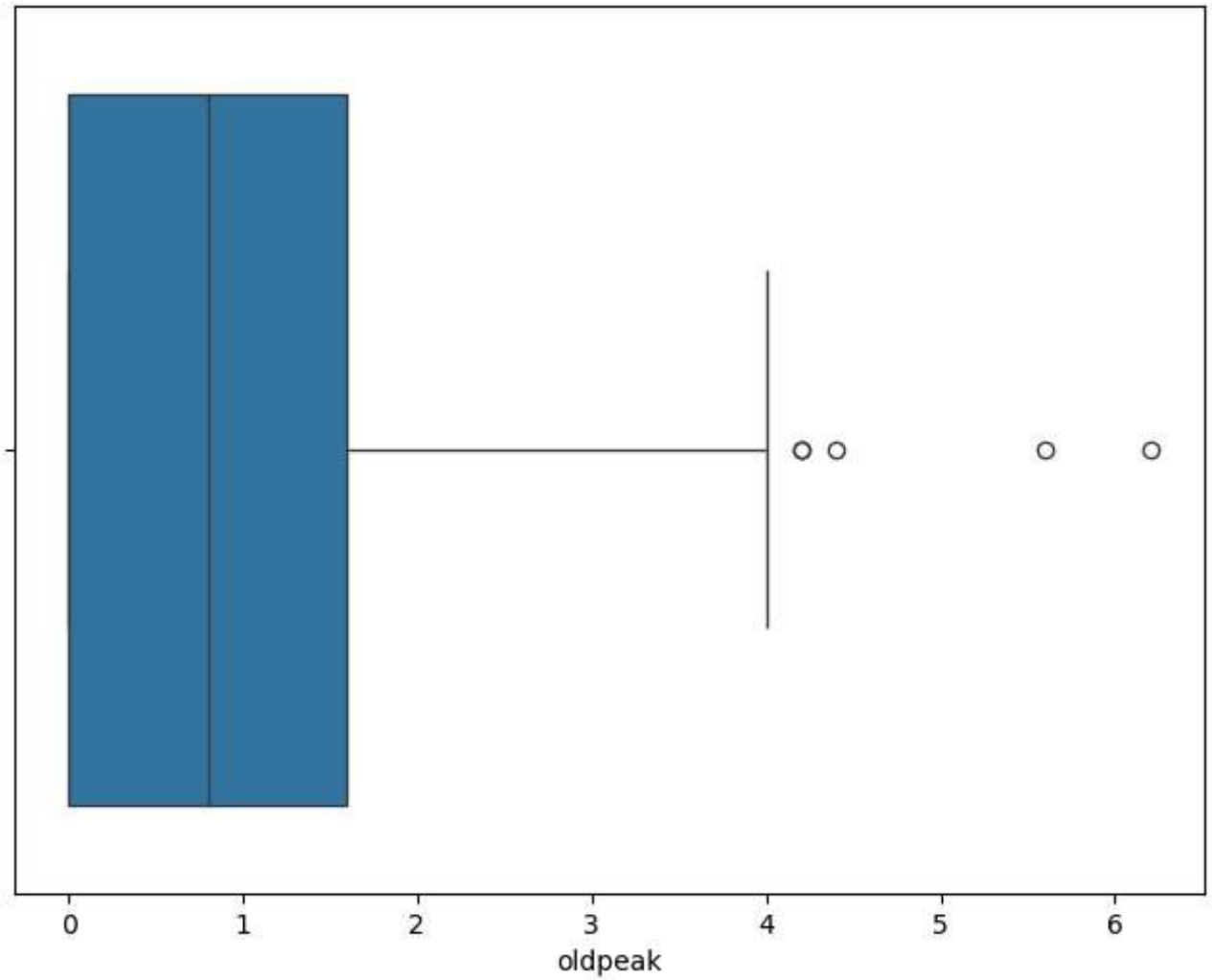
oldpeak variable

In [66]:
```
df['oldpeak'].describe()
```

Out[66]:
```
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
```

Box-plot of oldpeak variable

In [67]:
```
f, 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

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.

I hope you like this kernel on EDA journey.

Thanks