Exploratory Data Analysis (EDA) is a critical step in data analysis that involves using various techniques to understand the structure, patterns, relationships, and potential issues in a dataset before applying statistical models or machine learning algorithms. The goal of EDA is to explore and summarize the main characteristics of the data, often with visual methods, to gain insights and help guide further analysis.

Here are the key concepts in EDA theory:

1. Understanding the Dataset

1.Data Types: EDA begins by understanding the types of variables in the dataset (e.g., numerical, categorical, ordinal, etc.). Each type will require different techniques for analysis.

2.Missing Values: Identifying missing data points and deciding how to handle them (e.g., removing, imputing, or leaving them as-is).

3.Outliers: Detecting data points that significantly differ from others, which may indicate errors, anomalies, or important variations in the data.

4.Data Distribution: Understanding the distribution of the data (e.g., normal, skewed, bimodal) helps in deciding which statistical techniques are appropriate.

2. Statistical Summaries

1.Central Tendency: Measures such as the mean, median, and mode are used to understand the "center" of the data.

2.Dispersion: Measures such as variance, standard deviation, and interquartile range (IQR) help assess how spread out the data is.

3.Shape of Distribution: Skewness (asymmetry) and kurtosis (peakedness) help characterize the distribution of the data.

4.Correlations: Identifying relationships between variables using correlation coefficients (e.g., Pearson or Spearman) to understand associations.

3. Visualization Techniques

1.Histograms: Used to visualize the distribution of a single variable (especially numerical data).

2.Box Plots: Visualize the spread and detect outliers, displaying the median, quartiles, and potential outliers.

3.Bar Charts: For categorical variables, bar charts show the frequency or proportion of each category.

4.Scatter Plots: Useful for visualizing relationships between two numerical variables.

5.Pair Plots: Help visualize relationships among multiple numerical variables simultaneously.

6.Heatmaps: Used to visualize correlation matrices or missing data patterns.

7.Violin Plots: Combine aspects of box plots and density plots to show the distribution of data.

4. Handling Missing Data

1.Deletion: Removing rows or columns with missing data if it's minimal.

2.Imputation: Replacing missing values with estimates, such as the mean, median, mode, or a more sophisticated imputation method (e.g., using algorithms).

3.Marking Missing: Creating a separate indicator column that marks where data is missing.

5. Feature Engineering

1.Transformations: Sometimes, raw data needs to be transformed (e.g., scaling, normalization, or log transformations) to make it suitable for modeling.

2.Encoding Categorical Variables: Categorical data might be encoded into numerical forms (e.g., one-hot encoding, label encoding) for use in machine learning algorithms.

6. Dimensionality Reduction

.If the dataset has many features, dimensionality reduction techniques (such as Principal Component Analysis, PCA) may be applied to reduce complexity and highlight key features.

7. Identifying Patterns and Relationships

1.Clustering: Grouping data points into clusters based on similarity. This helps identify natural groupings within the data.

2.Trend Detection: Looking for trends or patterns over time or across different subgroups (e.g., seasonal patterns in time series data)

3.Anomaly Detection: Identifying data points that deviate significantly from expected patterns (outliers).

8. Feature Selection

After performing EDA, you may want to focus on the most relevant features for modeling. This is typically done by considering correlations, feature importance scores, or domain knowledge.

9. Data Transformation

1.Scaling and Normalization: Ensuring that numerical variables are on the same scale (especially important for algorithms like k-nearest neighbors or gradient descent).

2.Log Transformations: Used to handle skewed data or make distributions more normal.

10. Communication of Findings

The results from EDA should be clearly communicated using graphs, tables, and concise summaries to inform further analysis and decision-making.

Tools and Libraries for EDA

There are several tools and libraries commonly used for EDA, especially in Python:

Pandas: For data manipulation and basic statistical summaries.

Matplotlib & Seaborn: For creating static visualizations like histograms, scatter plots, and box plots.

Plotly: For interactive visualizations.

Scikit-learn: For more advanced statistical techniques and dimensionality reduction methods.

In [ ]:

```
# import libraries
import numpy as np
import pandas as pd


import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [65]:

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

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

In [67]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```python
# import dataset
```

In [69]:
```python
df = pd.read_csv(r"C:\Users\navee\OneDrive\Desktop\heart.csv")
```

In [71]:
```python
df
```

Out[71]:

	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

In [77]:
```python
#exploratory data analysis
print('The shape of the dataset : ', df.shape)
```

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

In [ ]:

In [79]:
```python
df.head()
```

Out[79]:

	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 [81]:
```python
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 [83]:
```
df.dtypes
```

Out[83]:
```
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 [85]:
```
df.describe()
```

Out[85]:

	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	0.5258
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 [87]:
```
df.columns
```

Out[87]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [89]:
```
df['target'].nunique()
```

Out[89]:
```
2
```

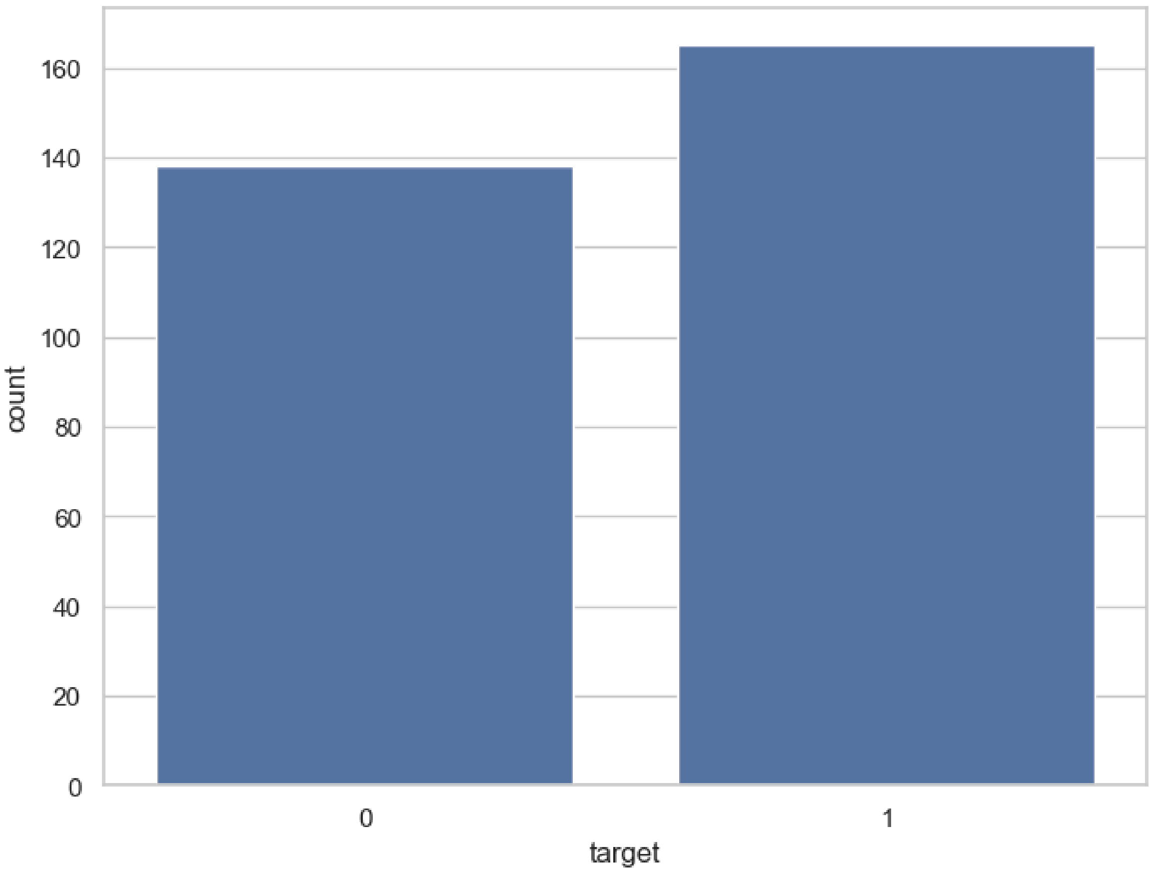
In [91]:
```
df['target'].nunique()
```

Out[91]:
```
2
```

In [93]:
```
df['target'].value_counts()
```

Out[93]:
```
target
1    165
0    138
Name: count, dtype: int64
```

In [125…
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=df)
plt.show()
```

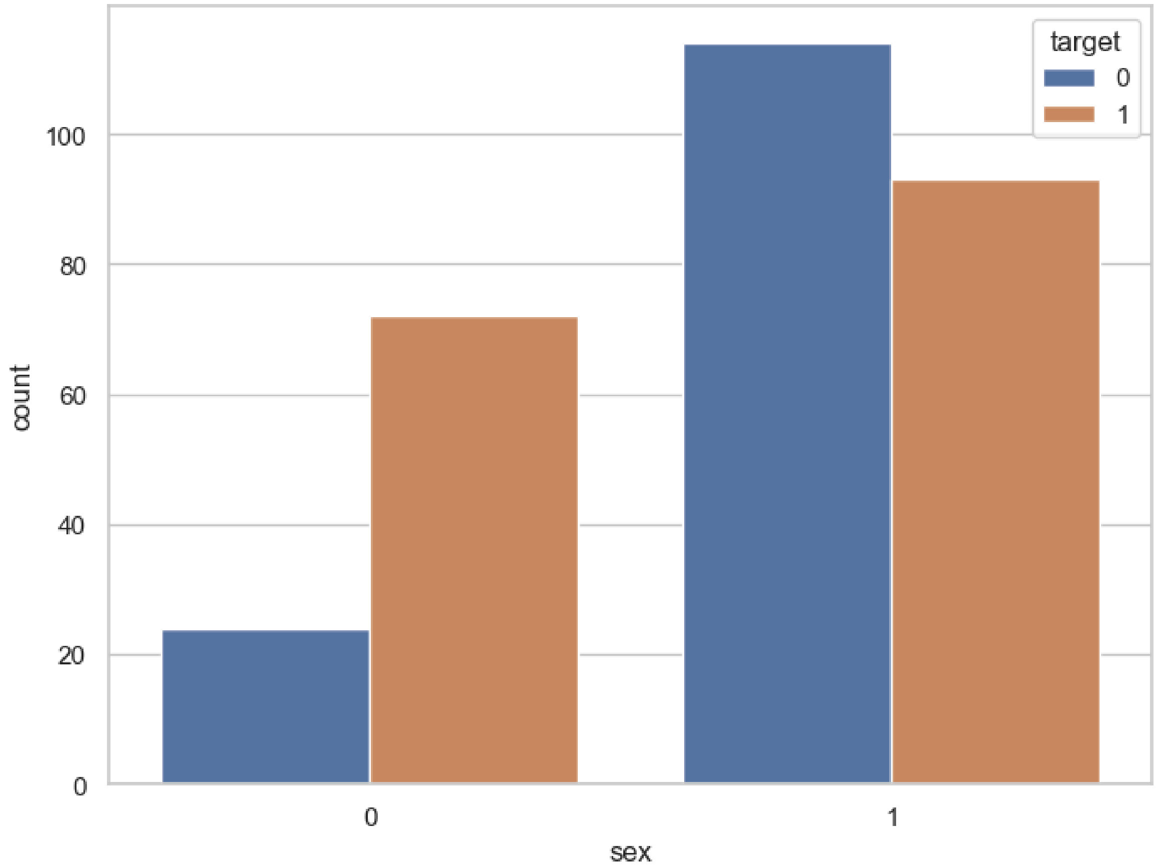


In [99]:
```
df.groupby('sex')['target'].value_counts()
```

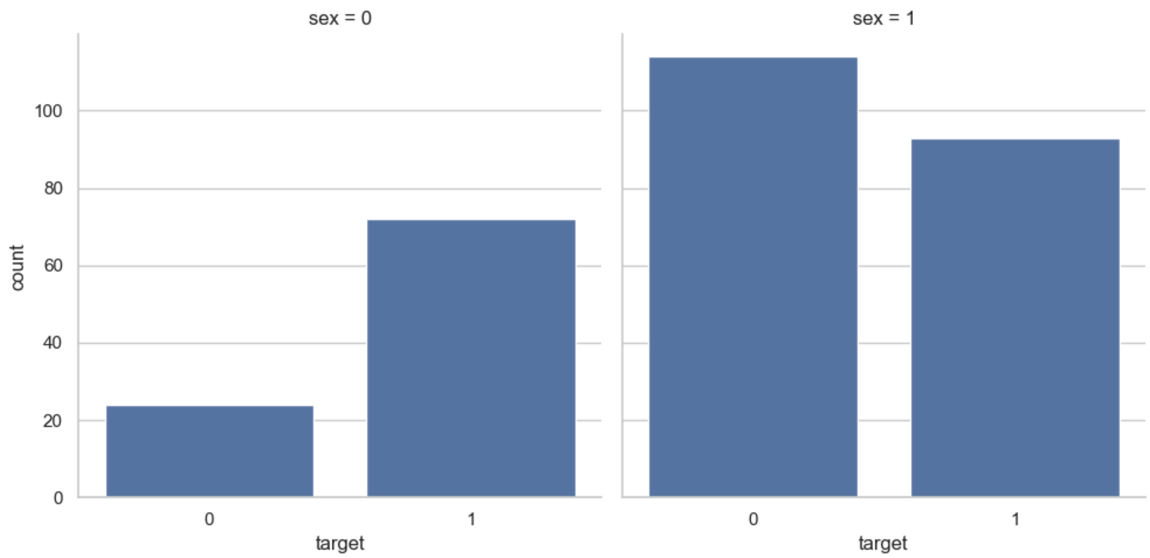
Out[99]:
```
sex  target
0    1          72
     0          24
1    0         114
     1          93
Name: count, dtype: int64
```

In [103…
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="sex", hue="target", data=df)
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

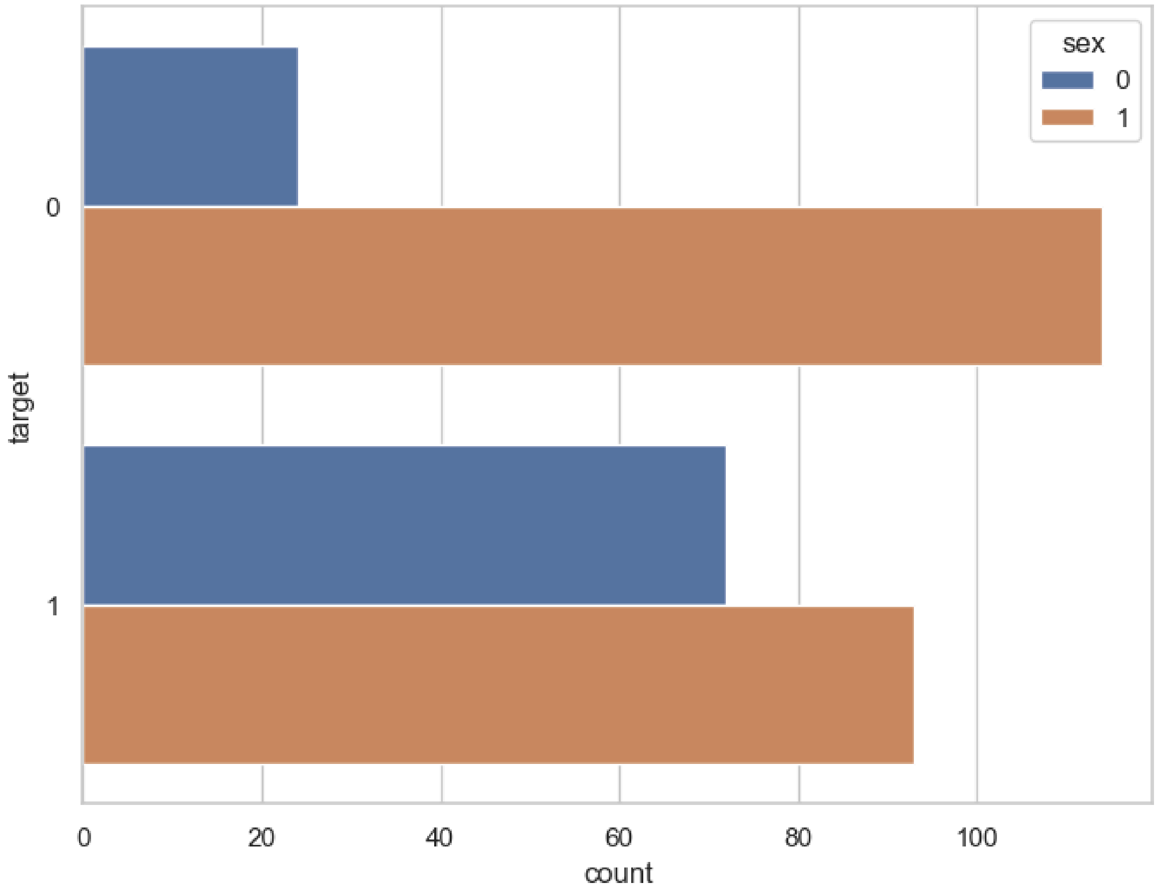


In [127…
```
ax = sns.catplot(x="target", col="sex", data=df, kind="count", height=5, aspect=
plt.show()
```



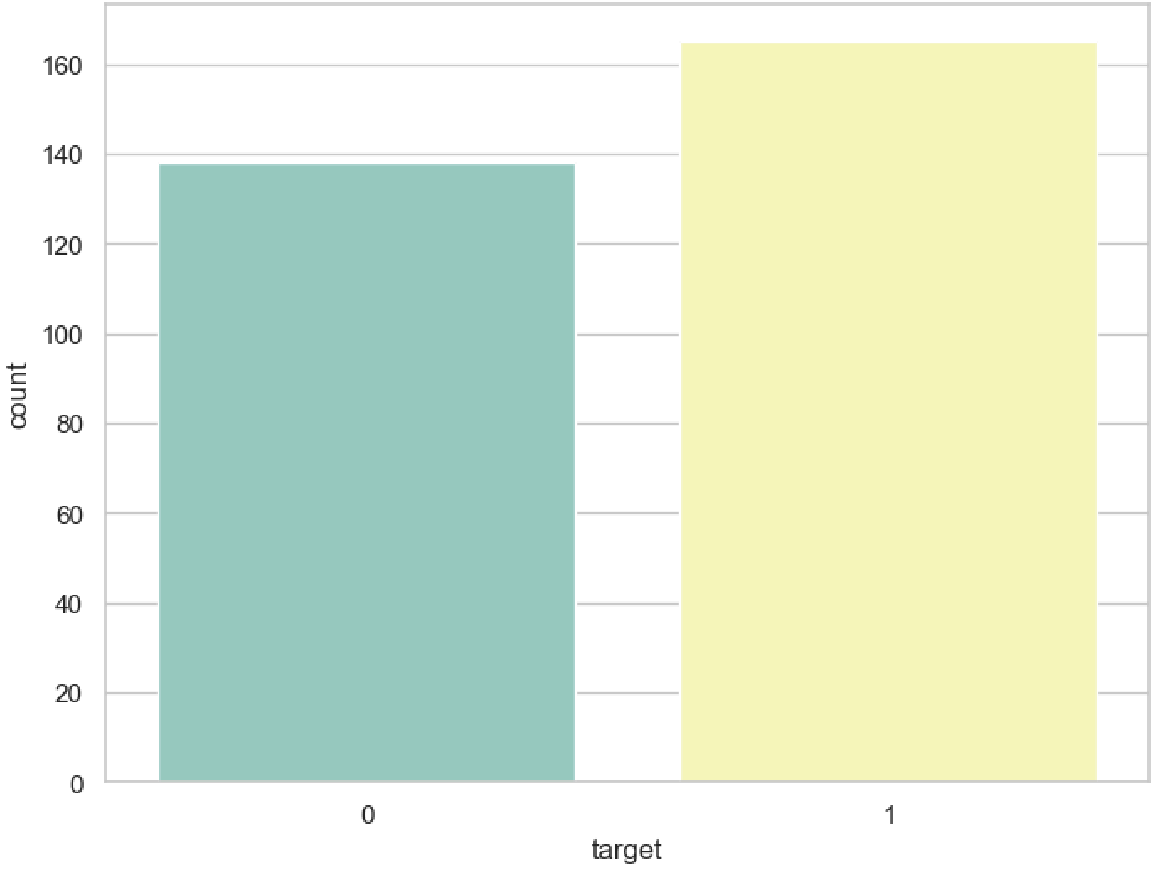
In [129…

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



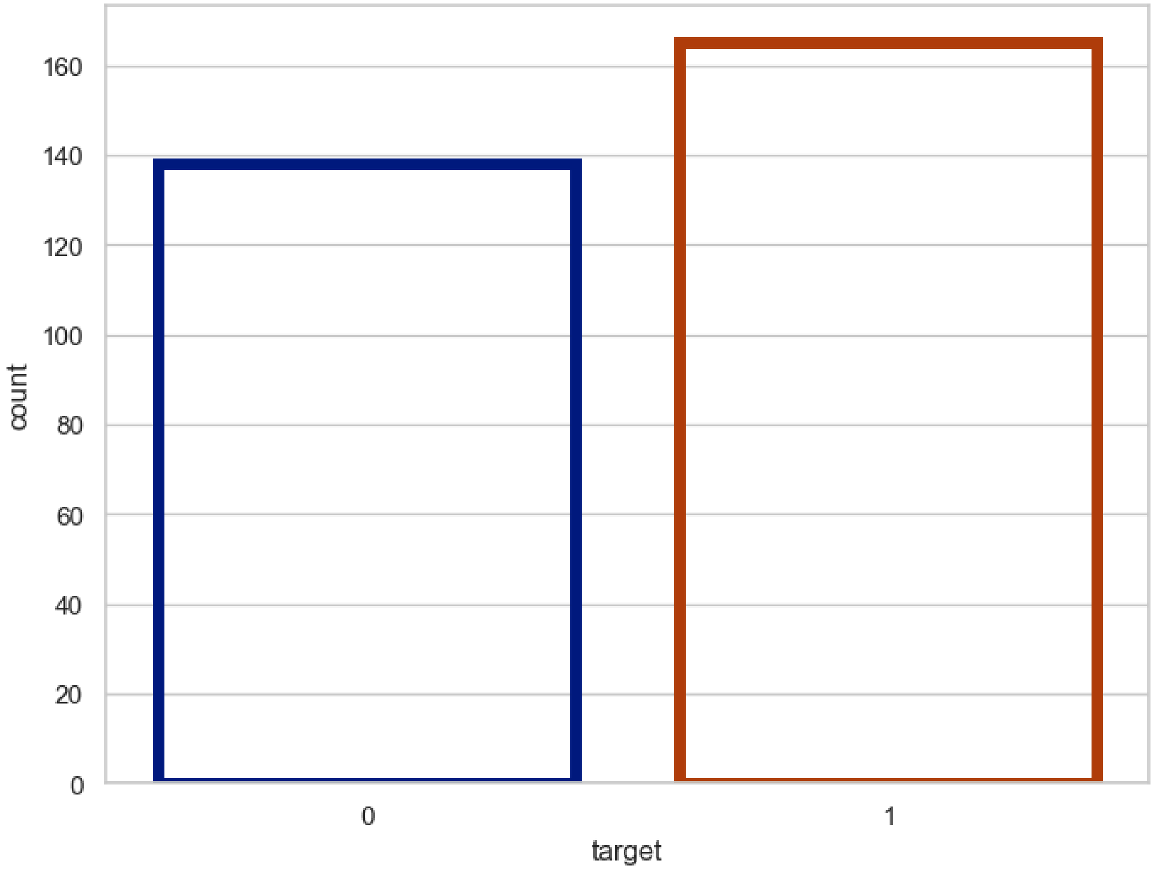
In [131…

```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=df, palette="Set3")
plt.show()
```



In [133…

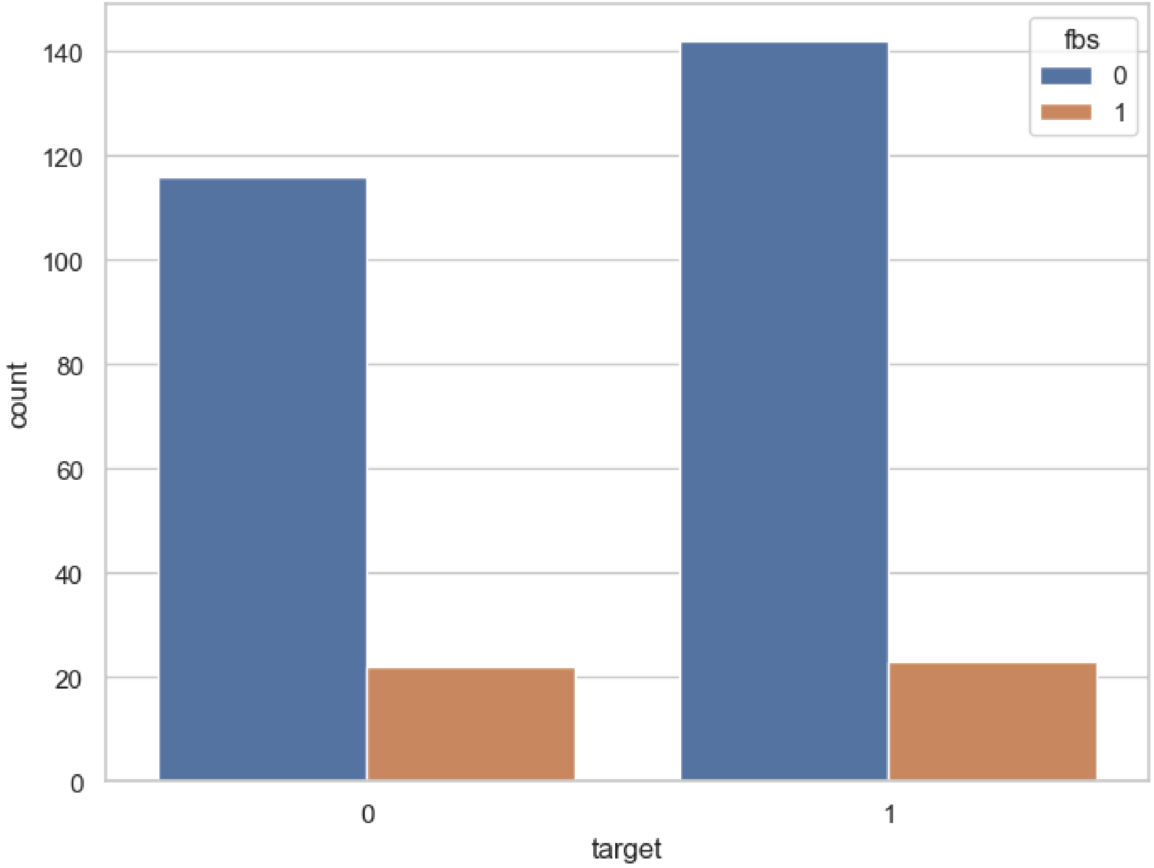
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=df, facecolor=(0, 0, 0, 0), linewidth=5, edg
plt.show()
```



In [135…

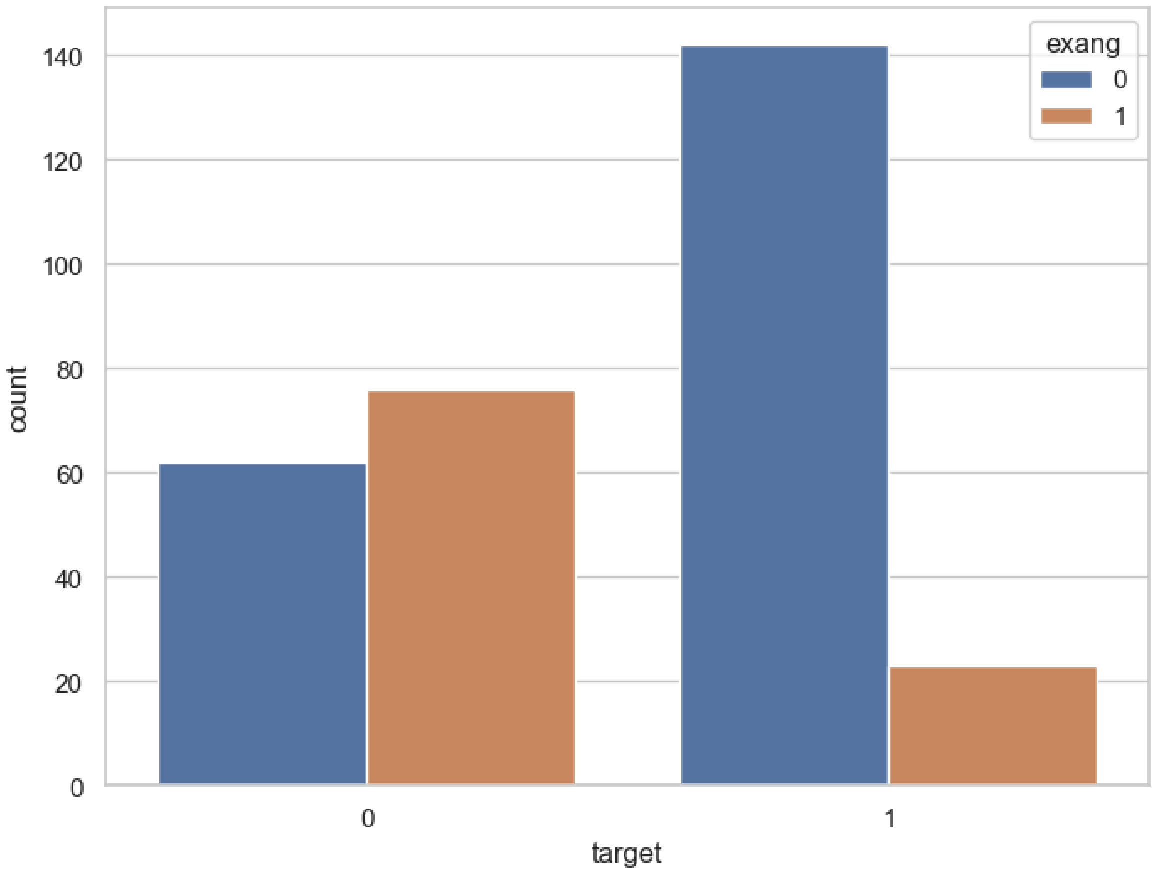
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", hue="fbs", data=df)
```

```
plt.show()
```



In [137…

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



In [139…
```
correlation = df.corr()
```

In [141…
```
correlation['target'].sort_values(ascending=False)
```

Out[141…
```
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
```

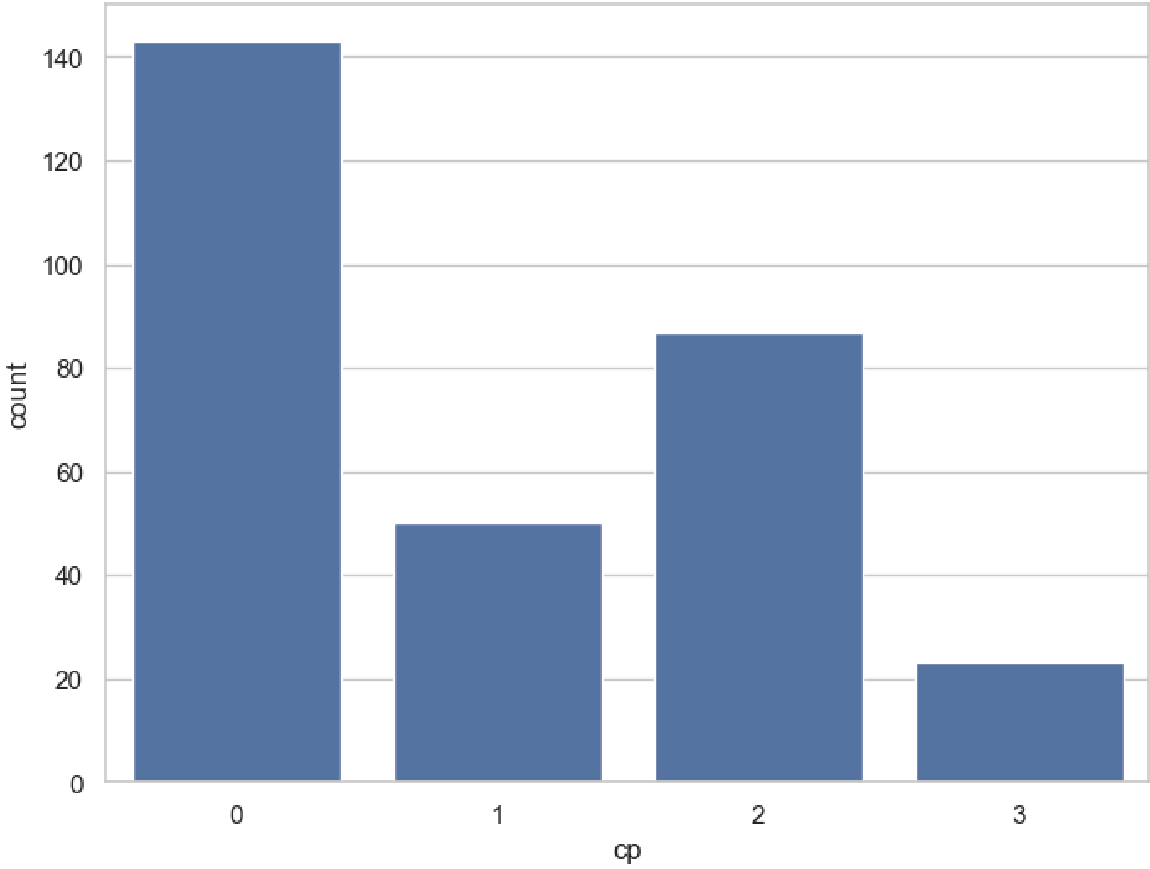
In [143…
```
df['cp'].nunique()
```

Out[143…
```
4
```

In [145…
```
df['cp'].value_counts()
```

Out[145…
```
cp
0    143
2     87
1     50
3     23
Name: count, dtype: int64
```

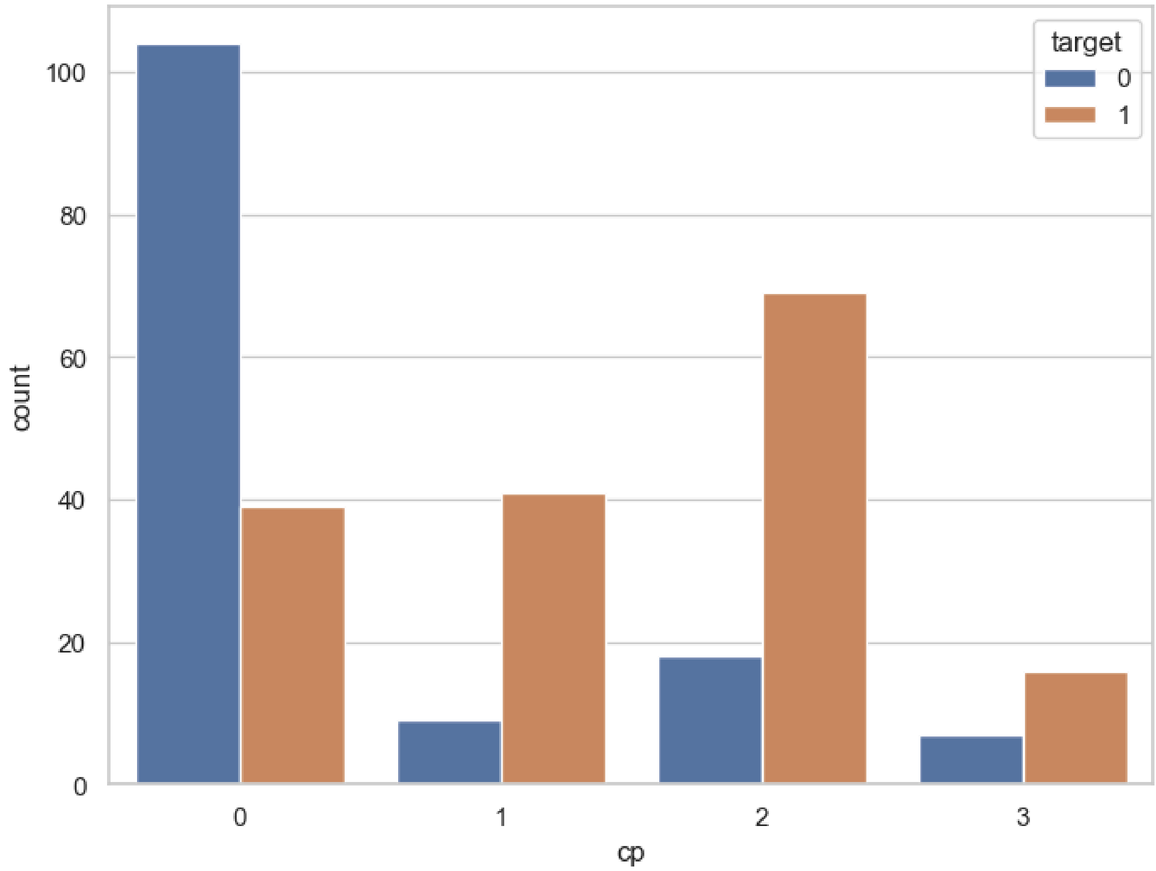
In [147…
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="cp", data=df)
plt.show()
```



In [149...
```
df.groupby('cp')['target'].value_counts()
```

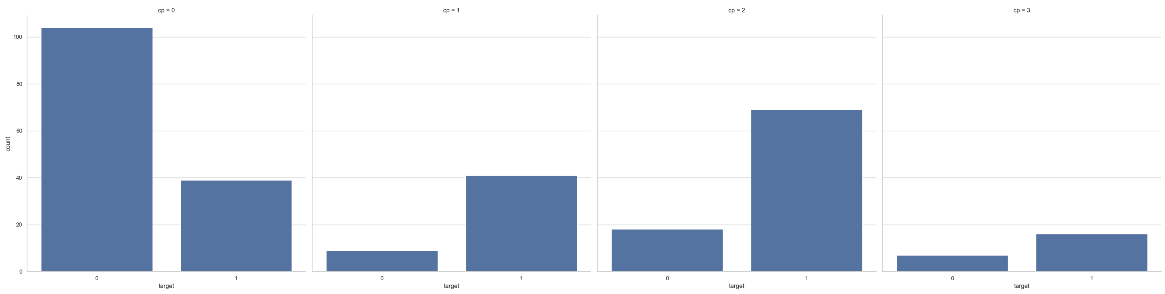
Out[149...
```
cp  target
0   0         104
    1          39
1   1          41
    0           9
2   1          69
    0          18
3   1          16
    0           7
Name: count, dtype: int64
```

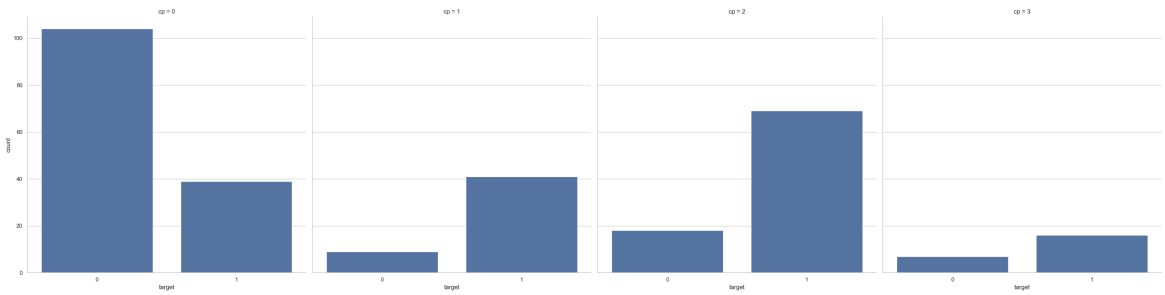
In [151...
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="cp", hue="target", data=df)
plt.show()
```



In [159…

```
ax = sns.catplot(x="target", col="cp", data=df, kind="count", height=8, aspect=1
plt.show()
```





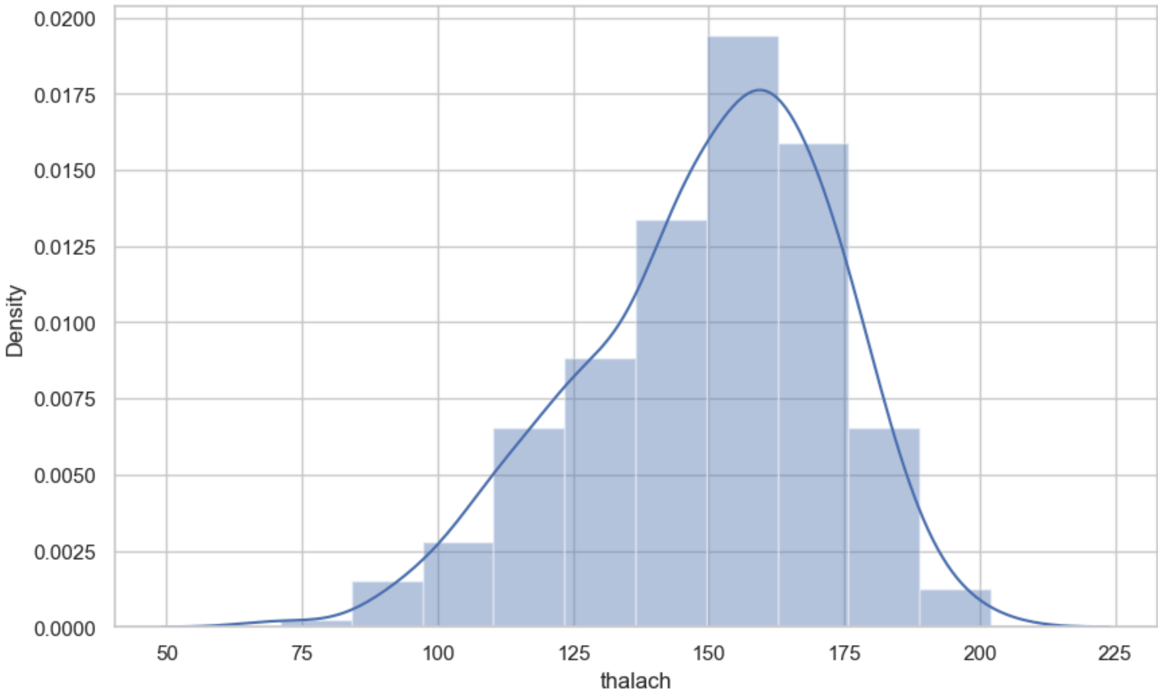
In [161…

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

Out[161… 91

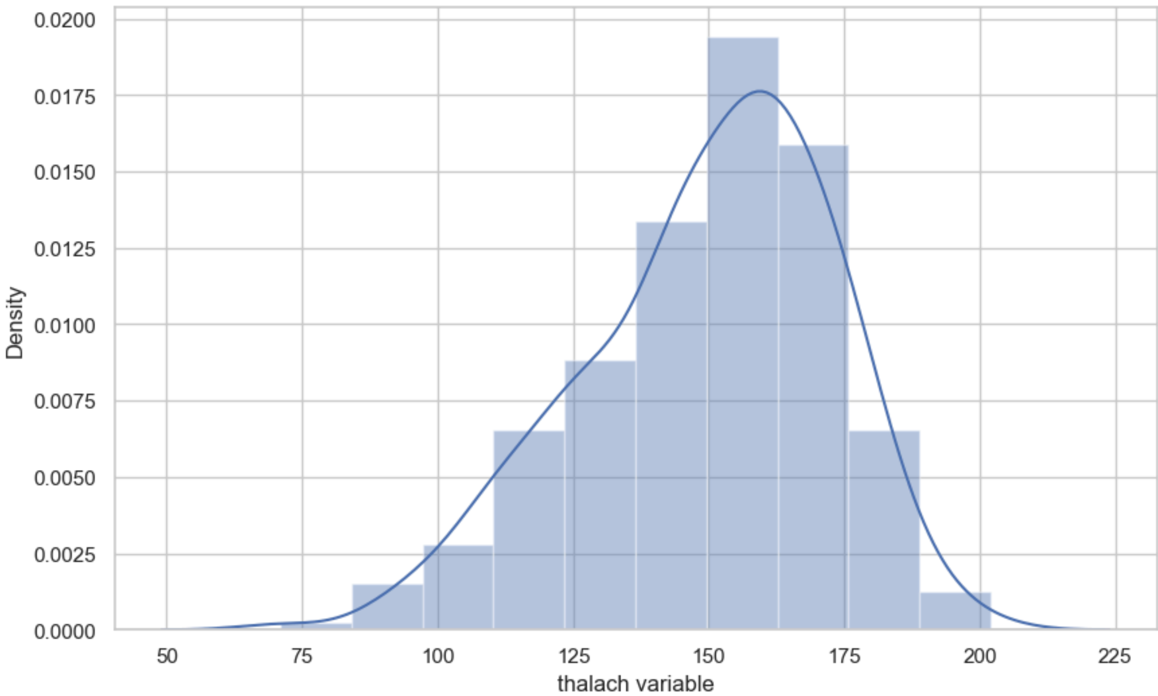
In [163…

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



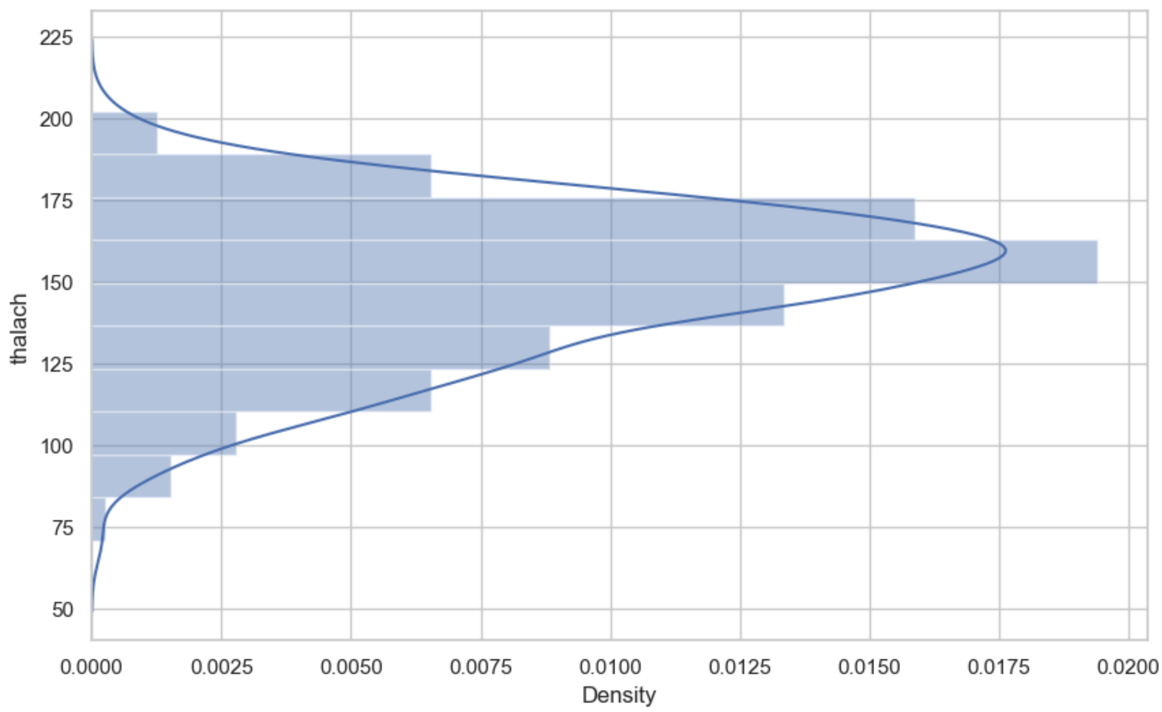
In [165…

```
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()
```



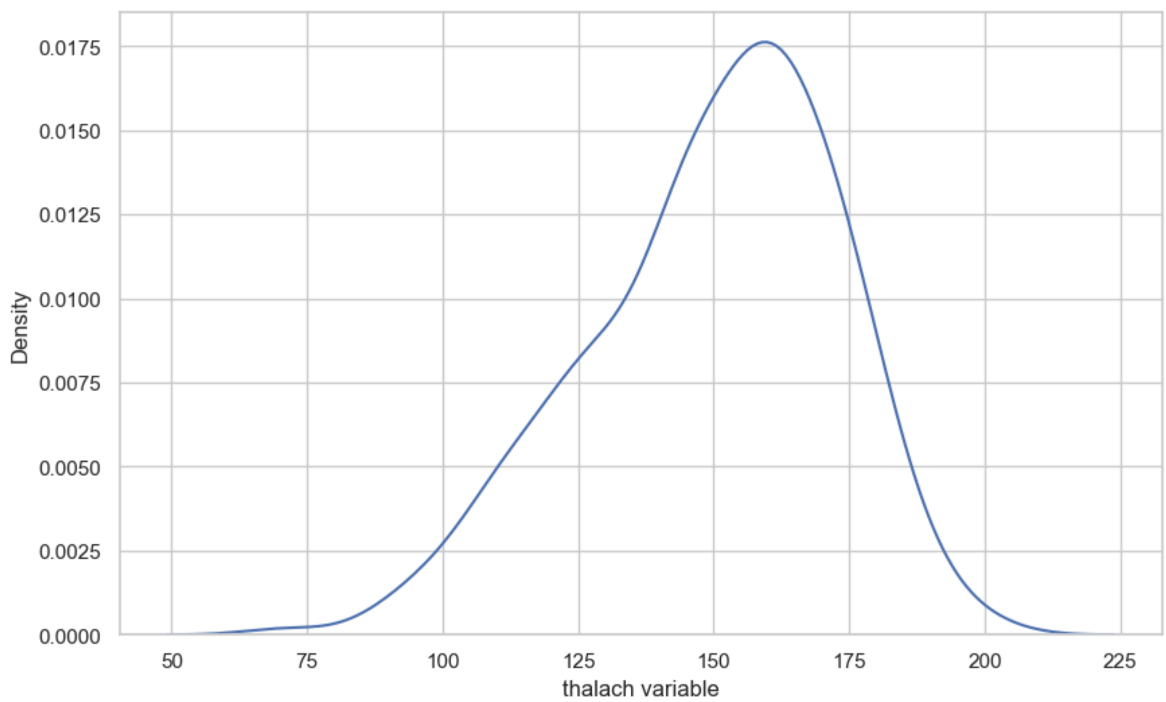
In [167…

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



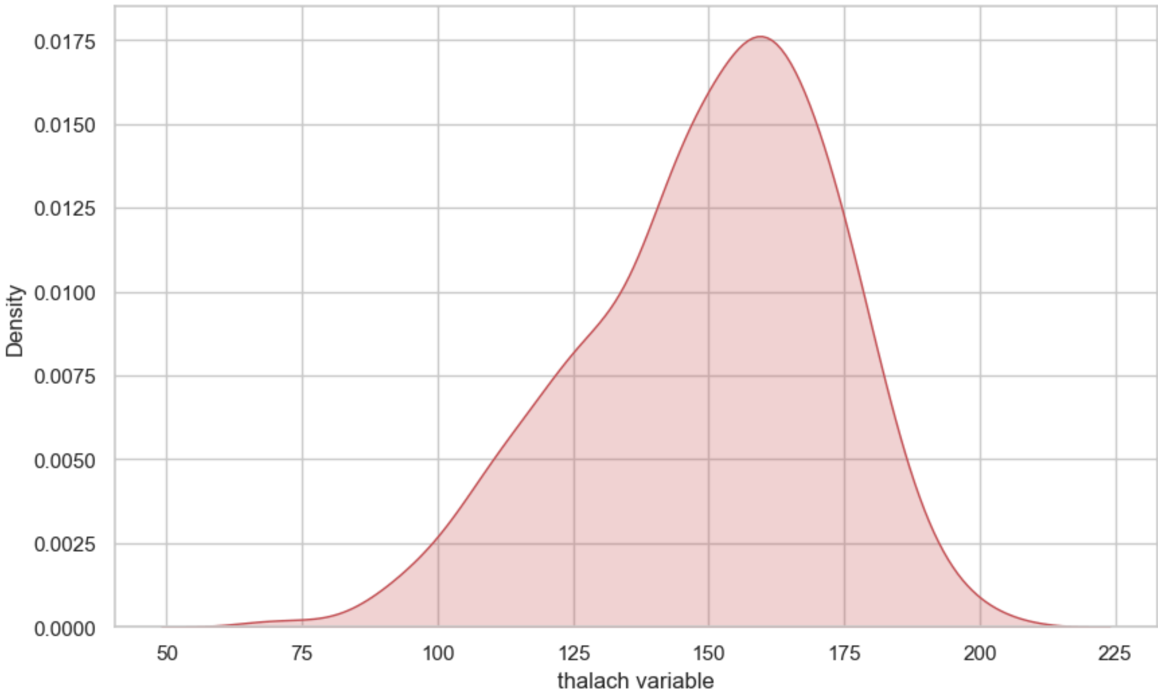
In [169…

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



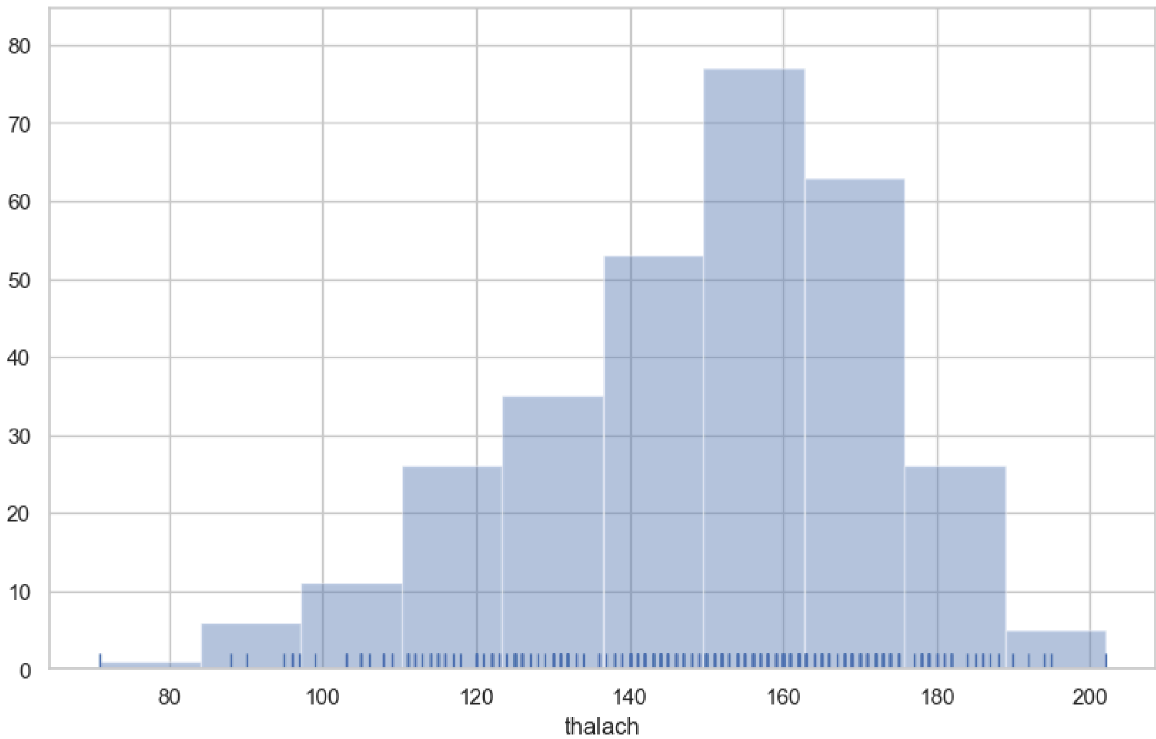
In [171…

```
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()
```



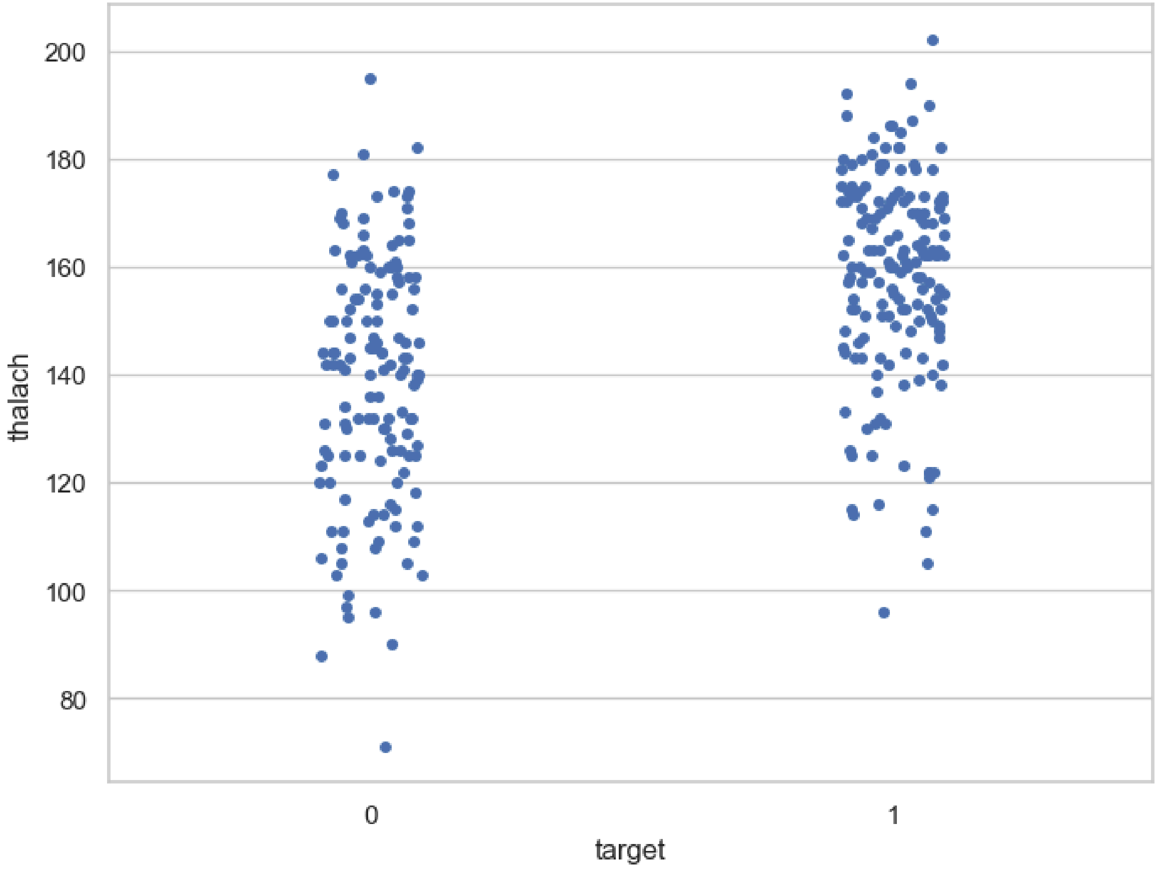
In [173…

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



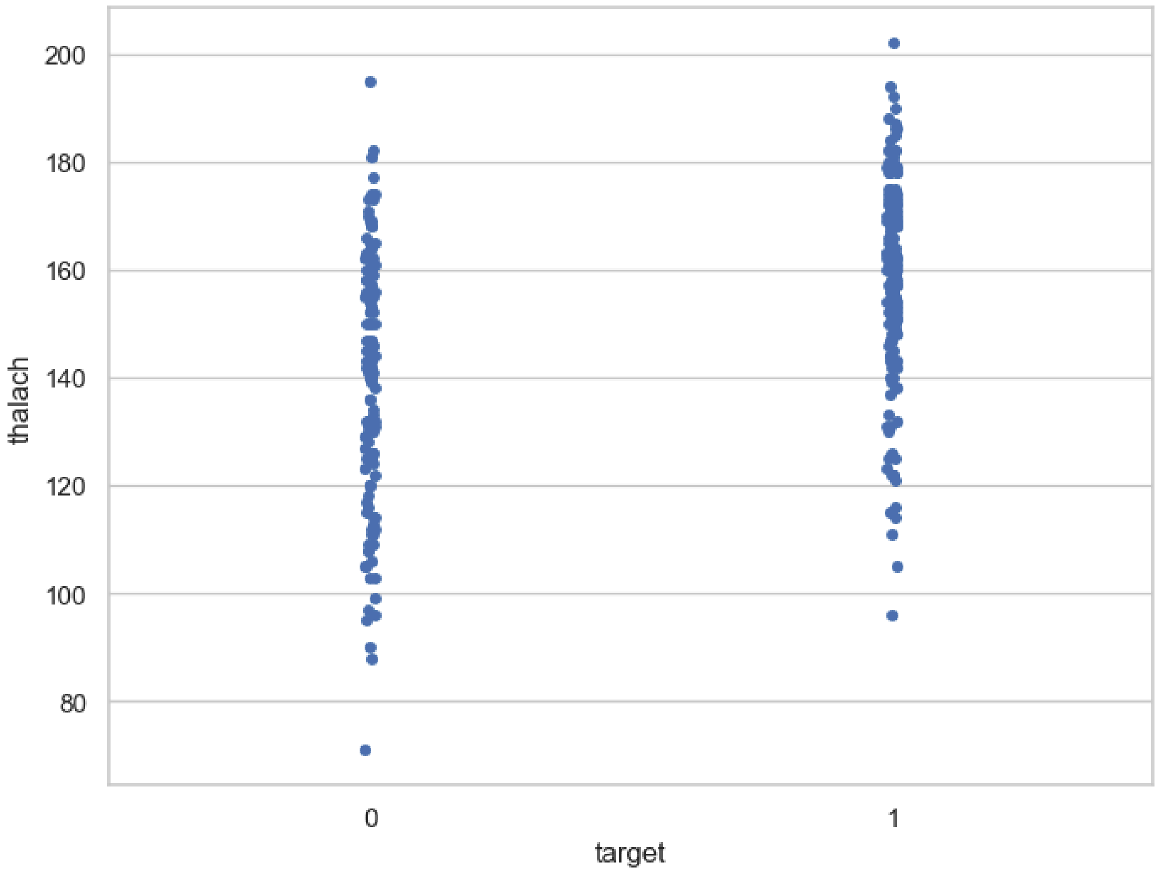
In [175…

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



In [177…

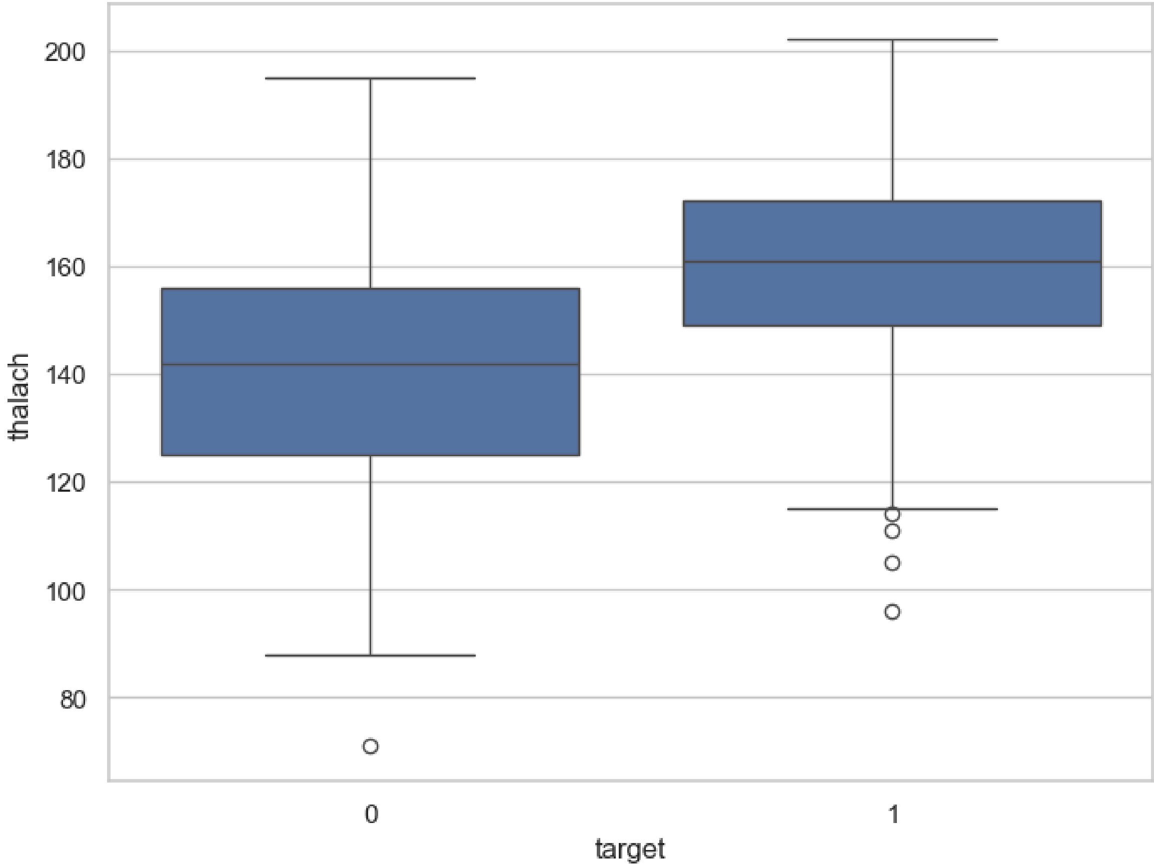
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="target", y="thalach", data=df, jitter = 0.01)
plt.show()
```



In [179…

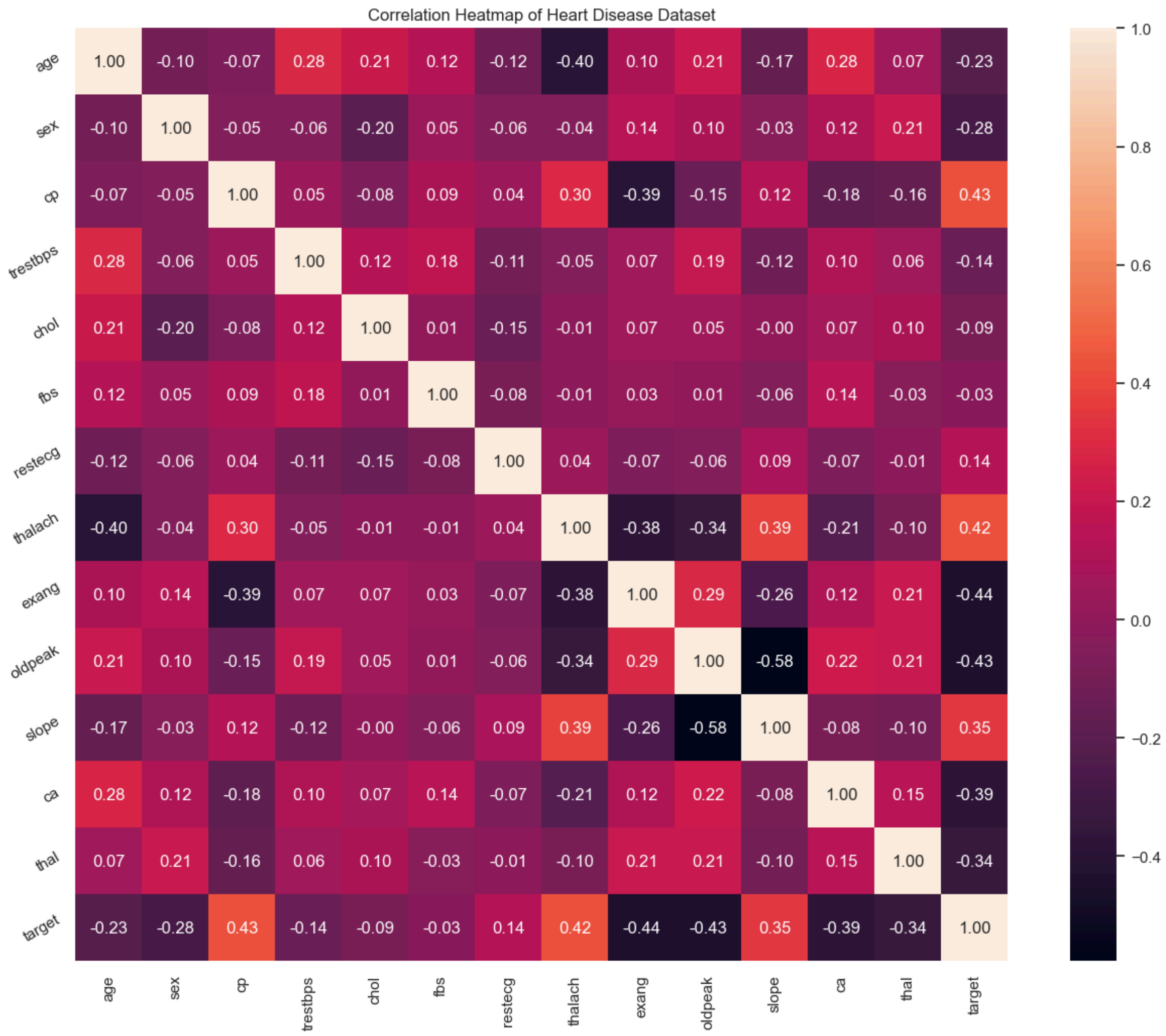
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x="target", y="thalach", data=df)
```

```
plt.show()
```



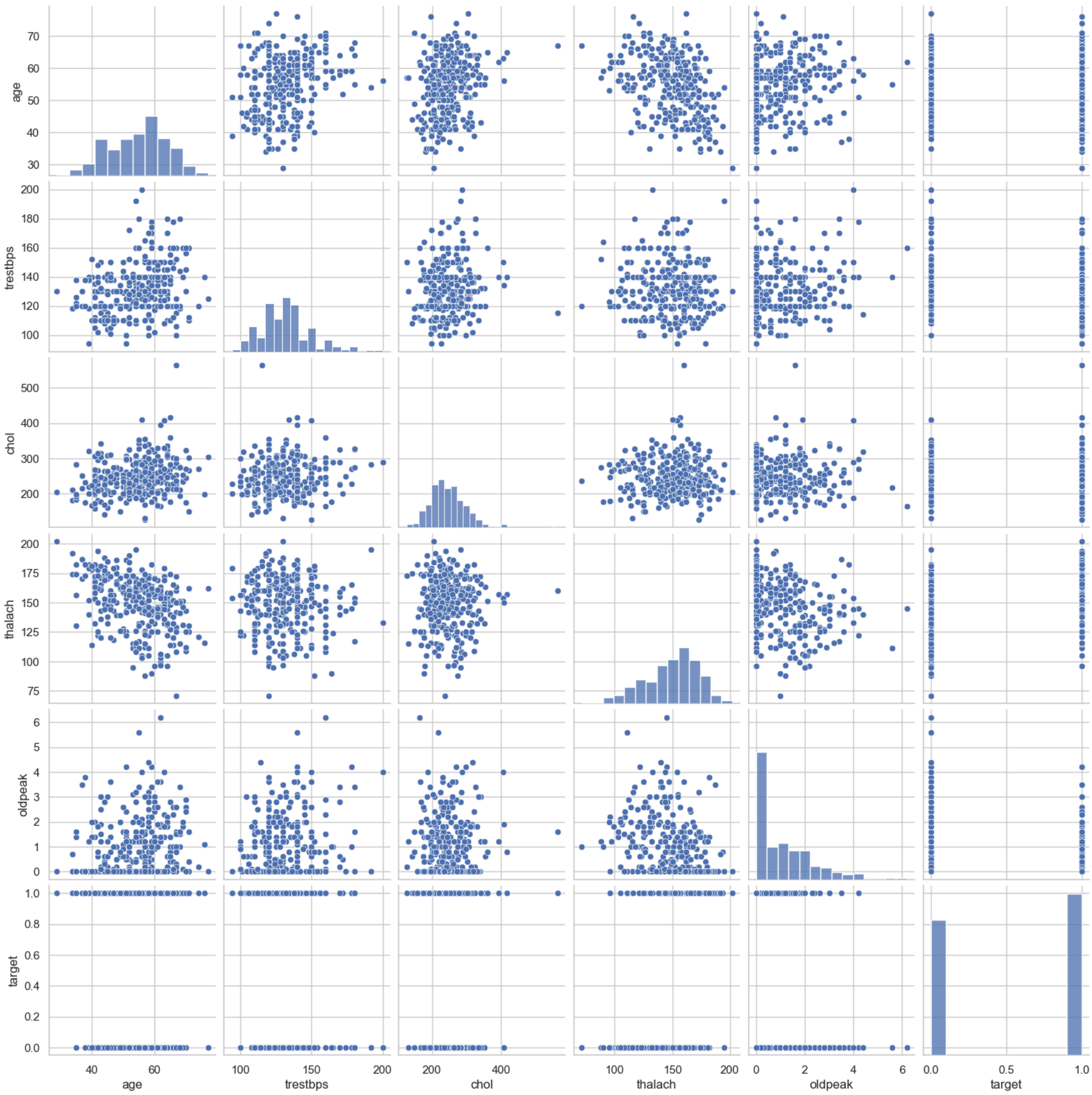
In [181…

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



In [184…

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



In [186…

```
df['age'].nunique()
```

Out[186…

```
41
```

In [188…

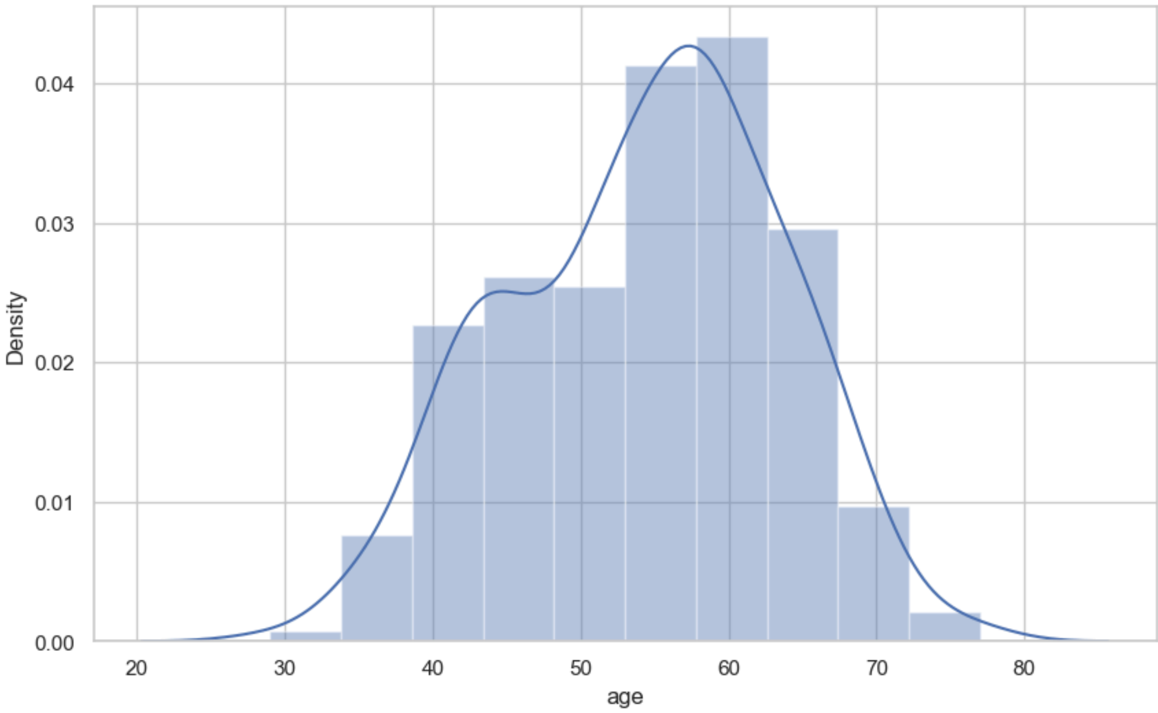
```
df['age'].describe()
```

Out[188…

```
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
```

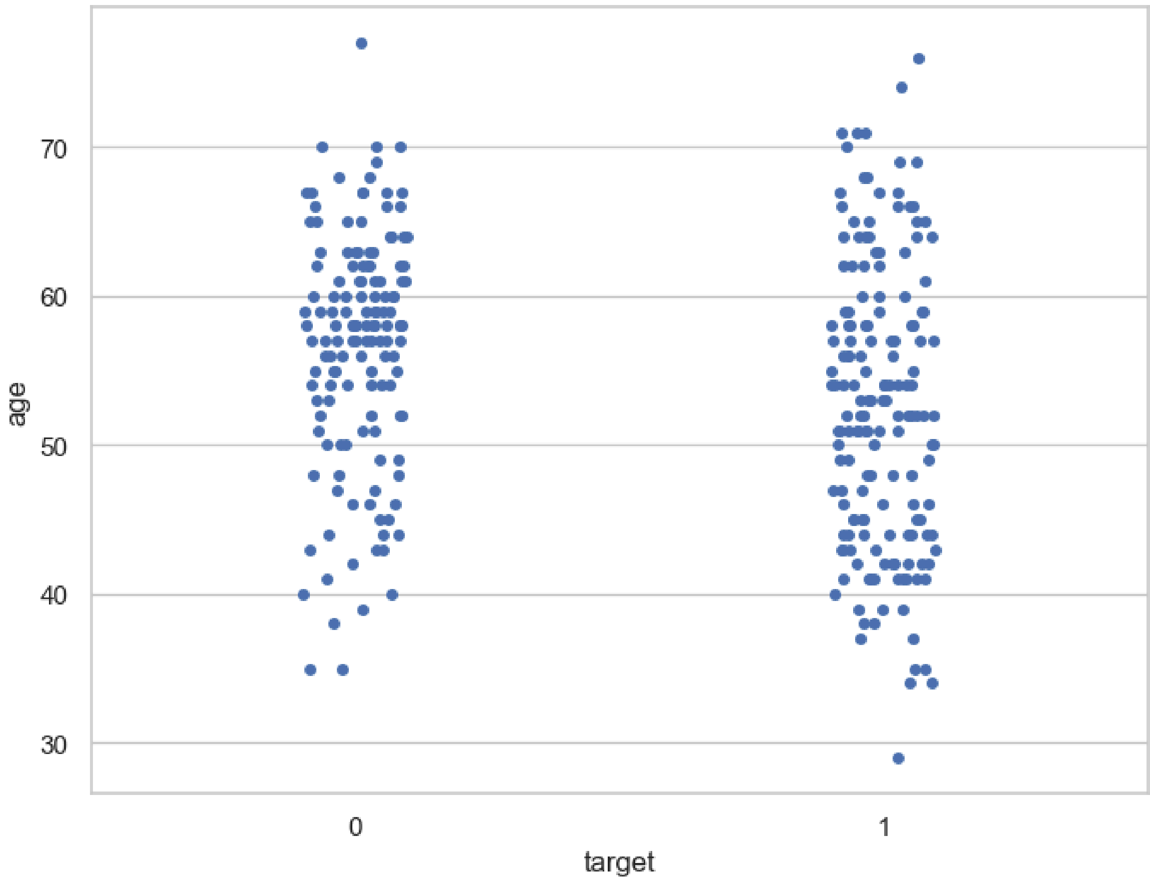
In [190…

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



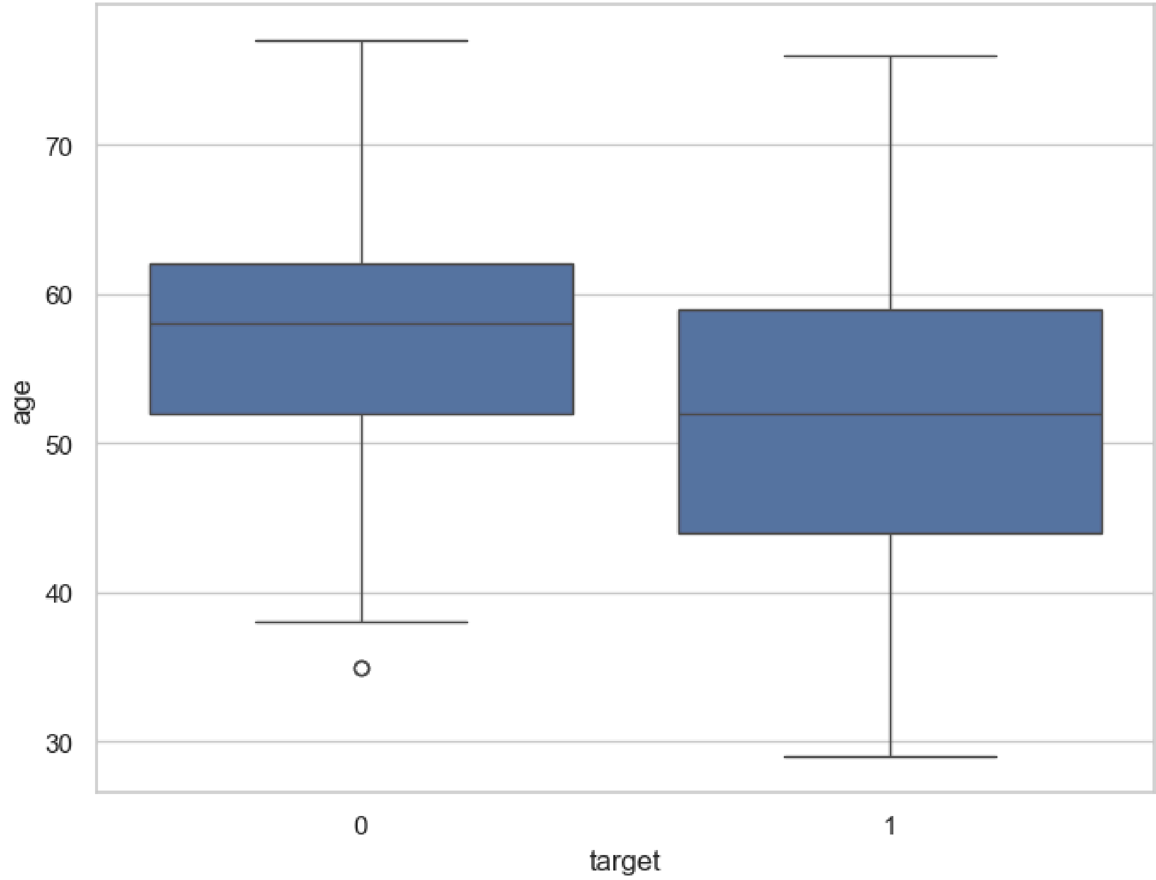
In [192…

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



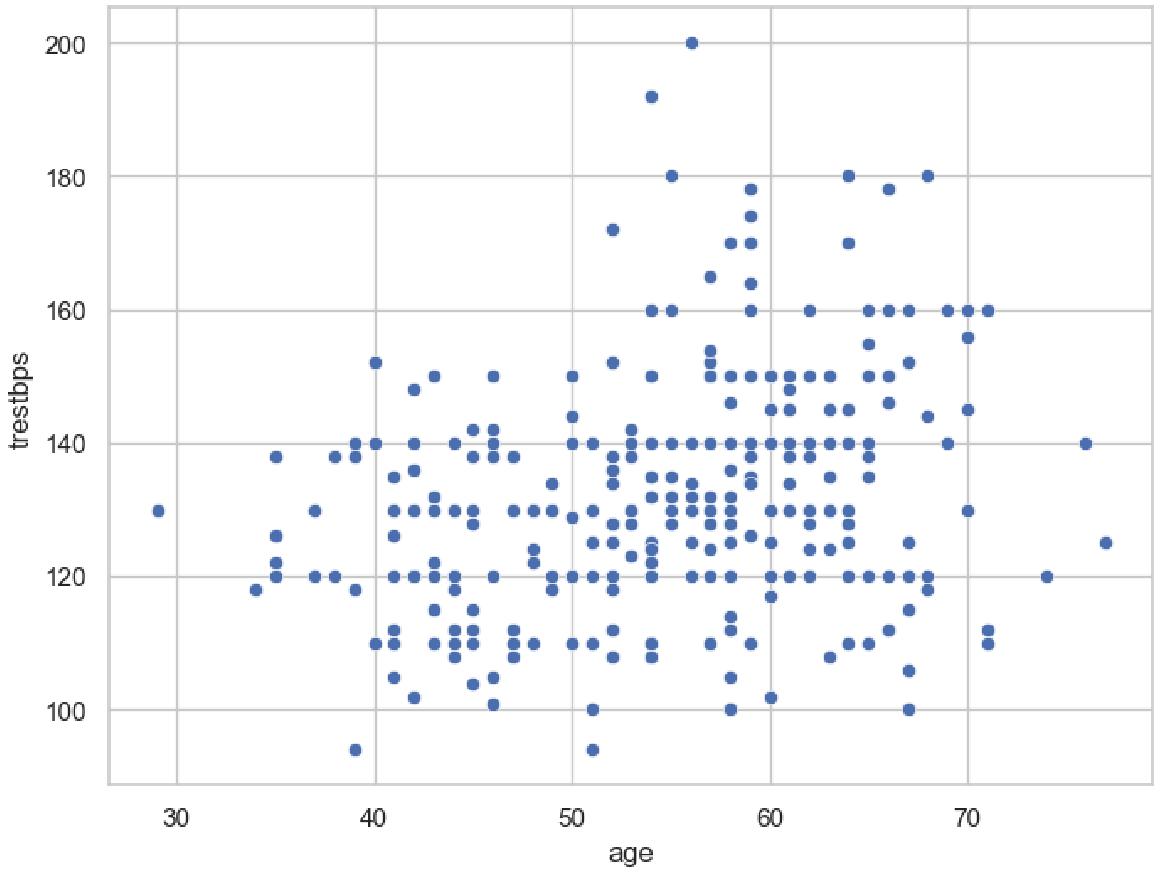
In [194…

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



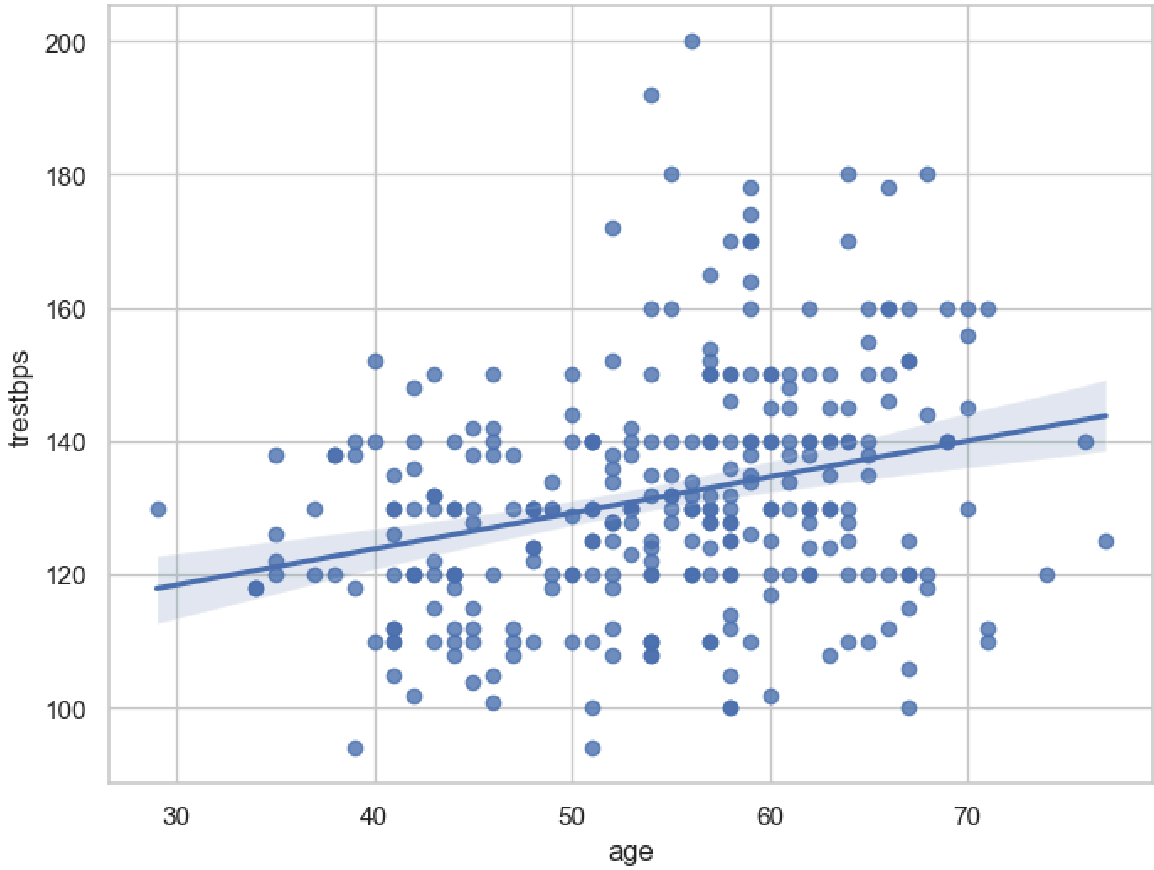
In [196…

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



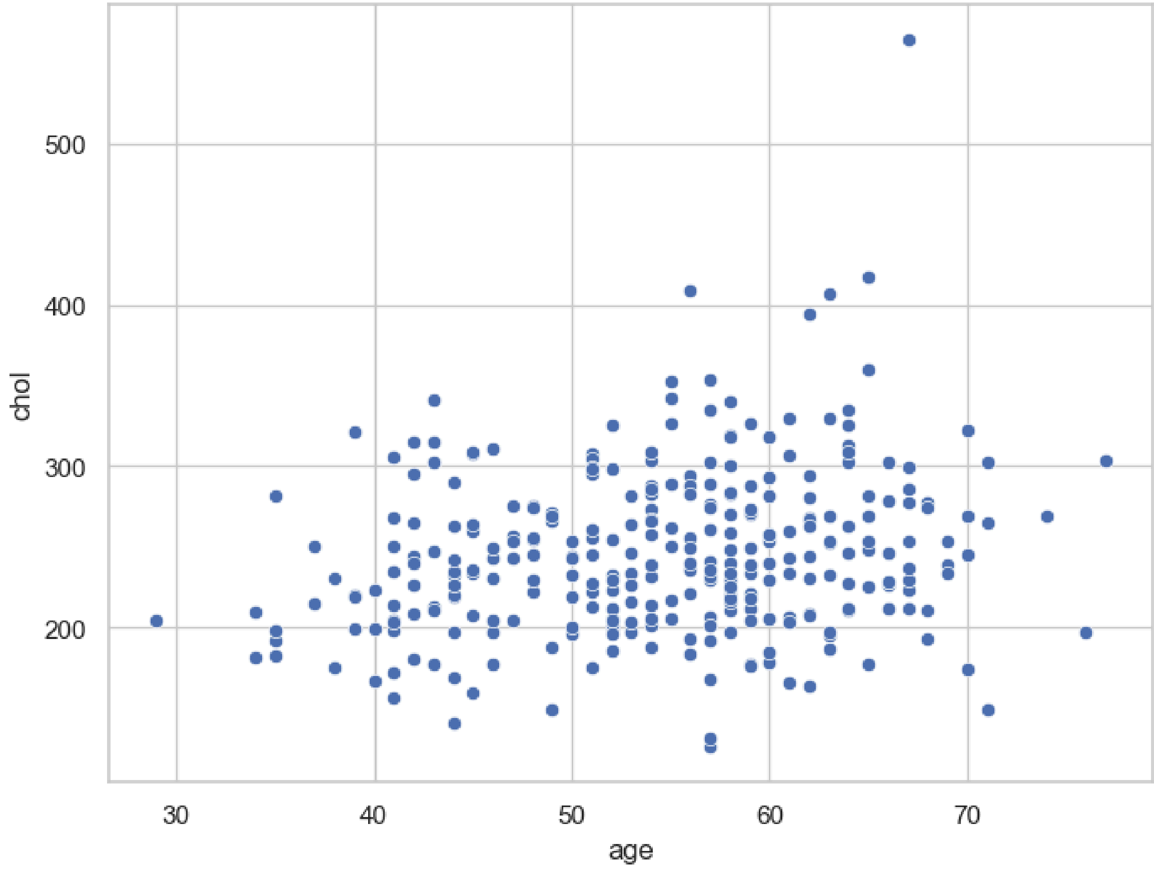
In [198…

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



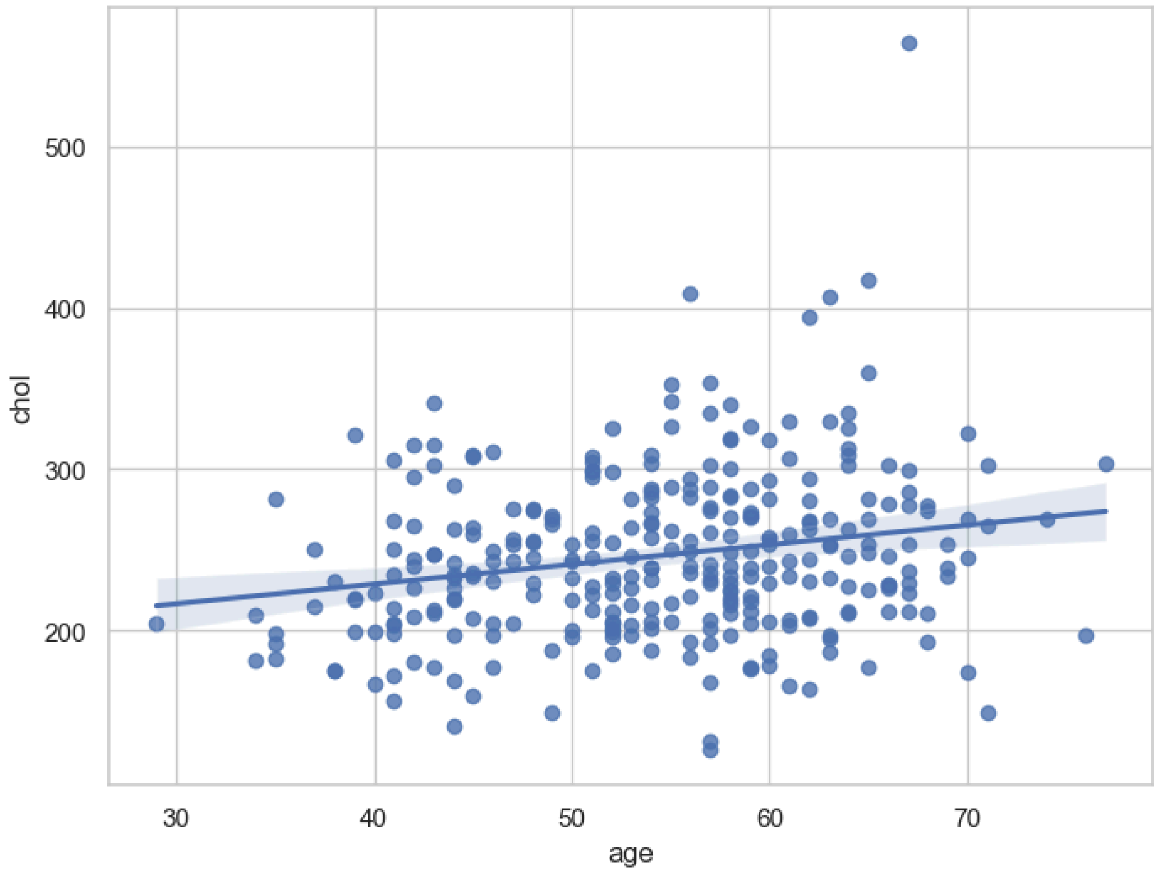
In [200…

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



In [202…

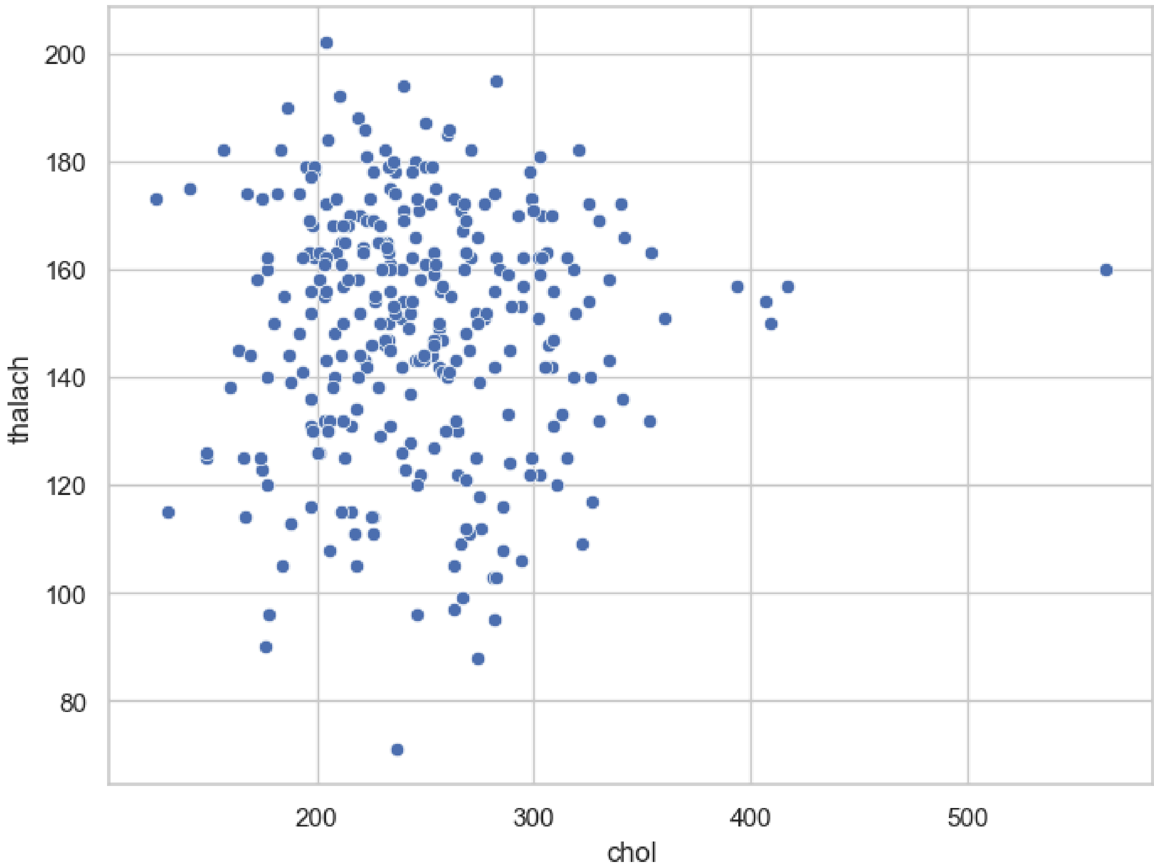
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="age", y="chol", data=df)
plt.show()
```



In [204…

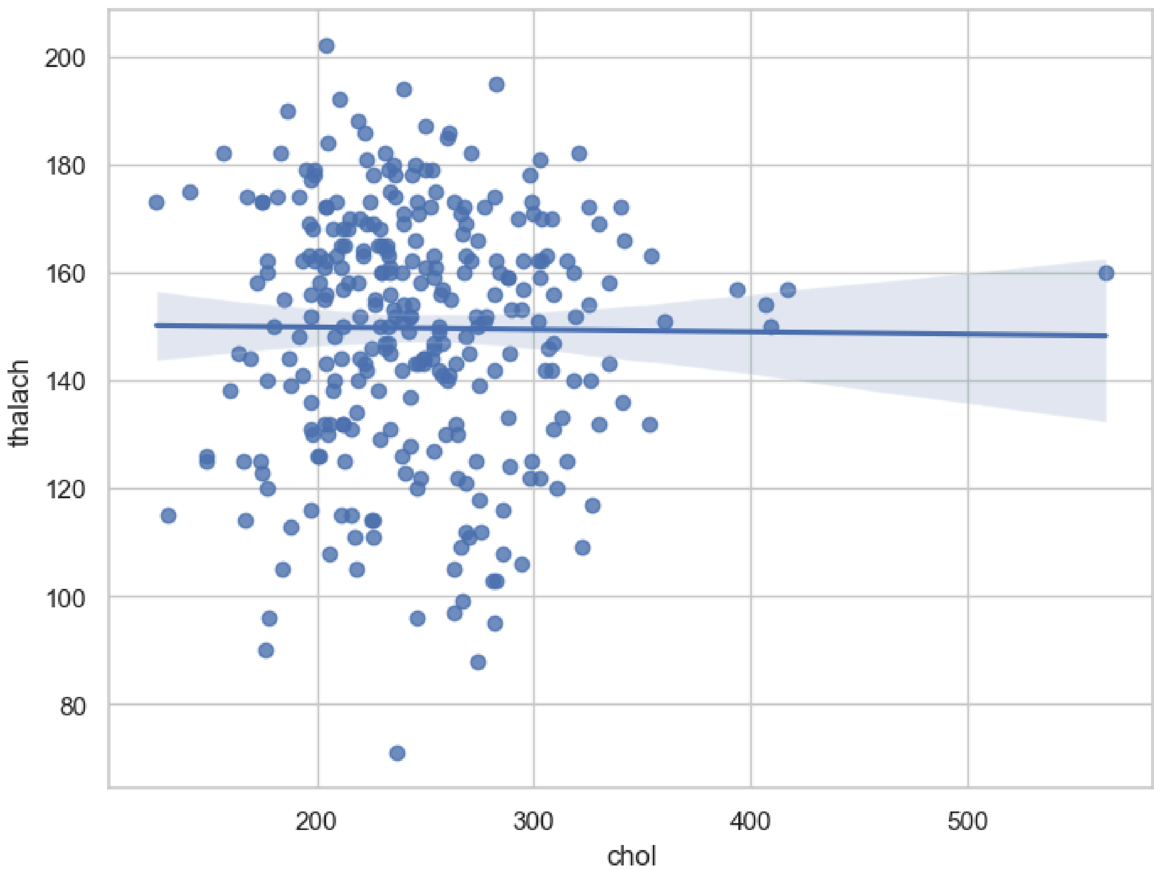
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.scatterplot(x="chol", y = "thalach", data=df)
```

```
plt.show()
```



In [206…

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



In [208…
```
df.isnull().sum()
```

Out[208…
```
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
```

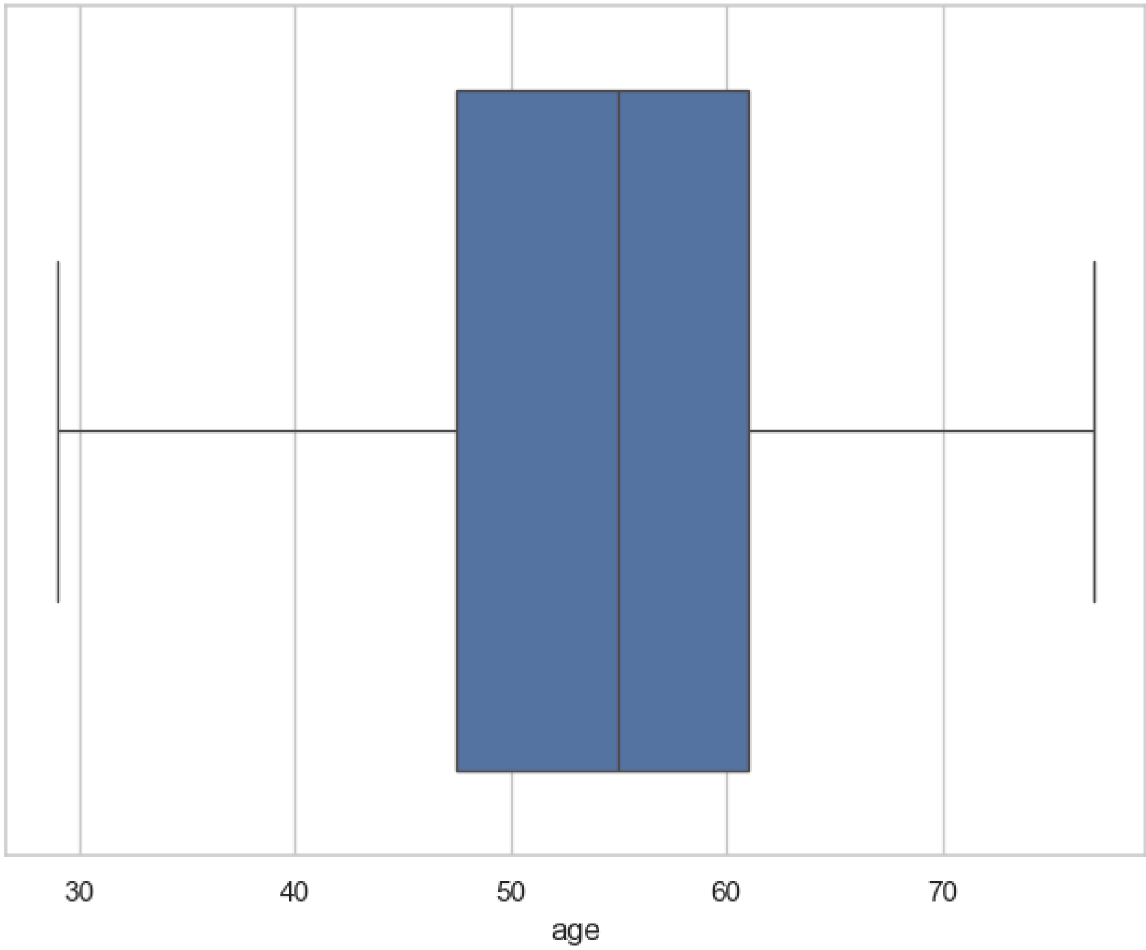
In [210…
```
assert pd.notnull(df).all().all()
```

In [212…
```
assert (df >= 0).all().all()
```

In [214…
```
df['age'].describe()
```

Out[214…
```
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
```

In [216…
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["age"])
plt.show()
```



In [218…

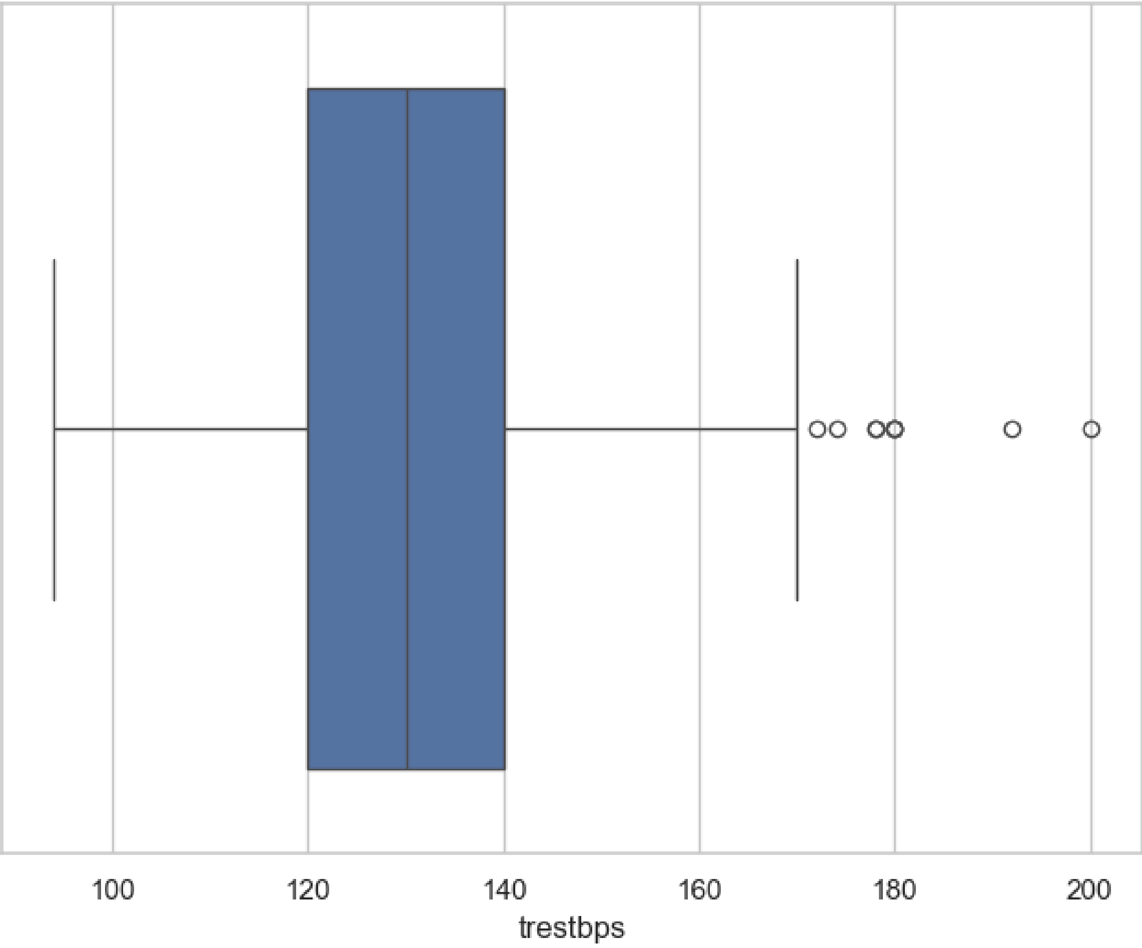
```
df['trestbps'].describe()
```

Out[218…

```
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 [220…

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



In [222…

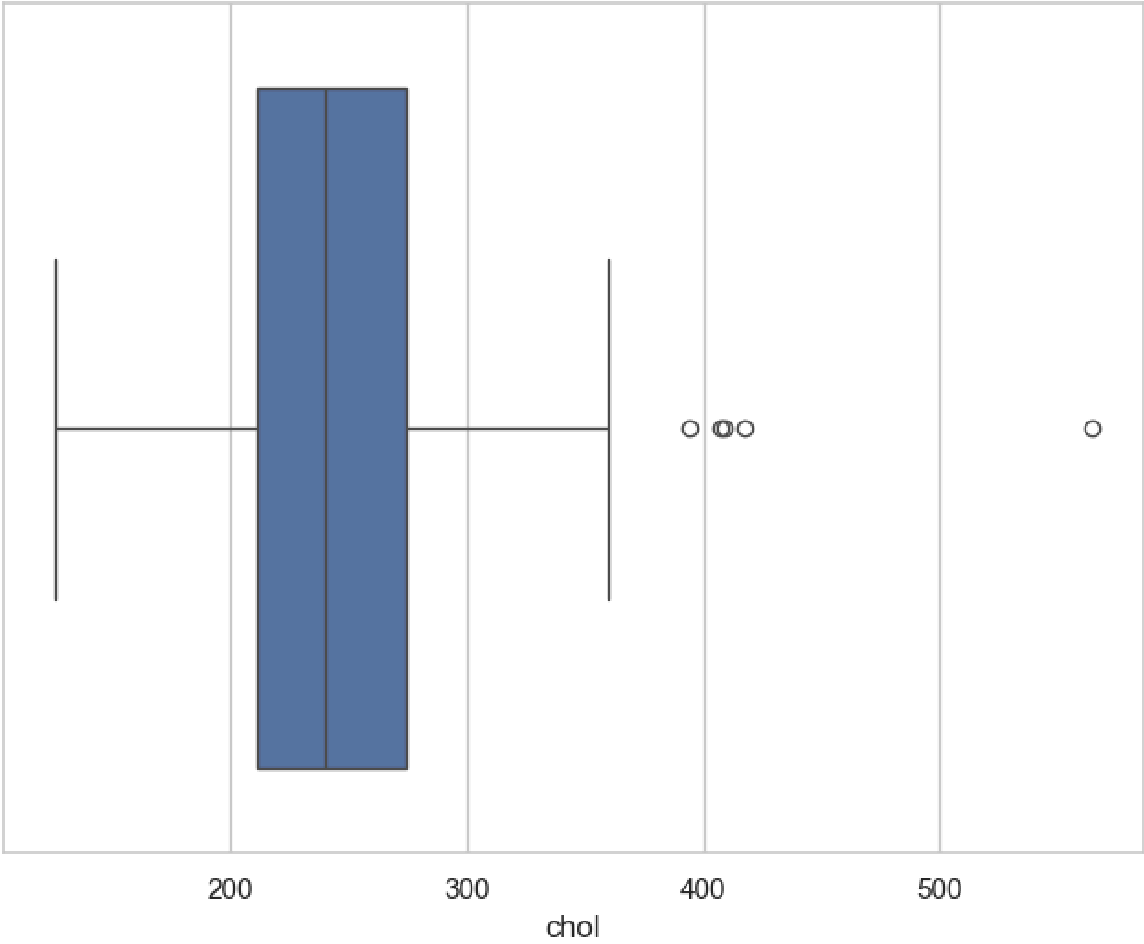
```
df['chol'].describe()
```

Out[222…

```
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 [224…

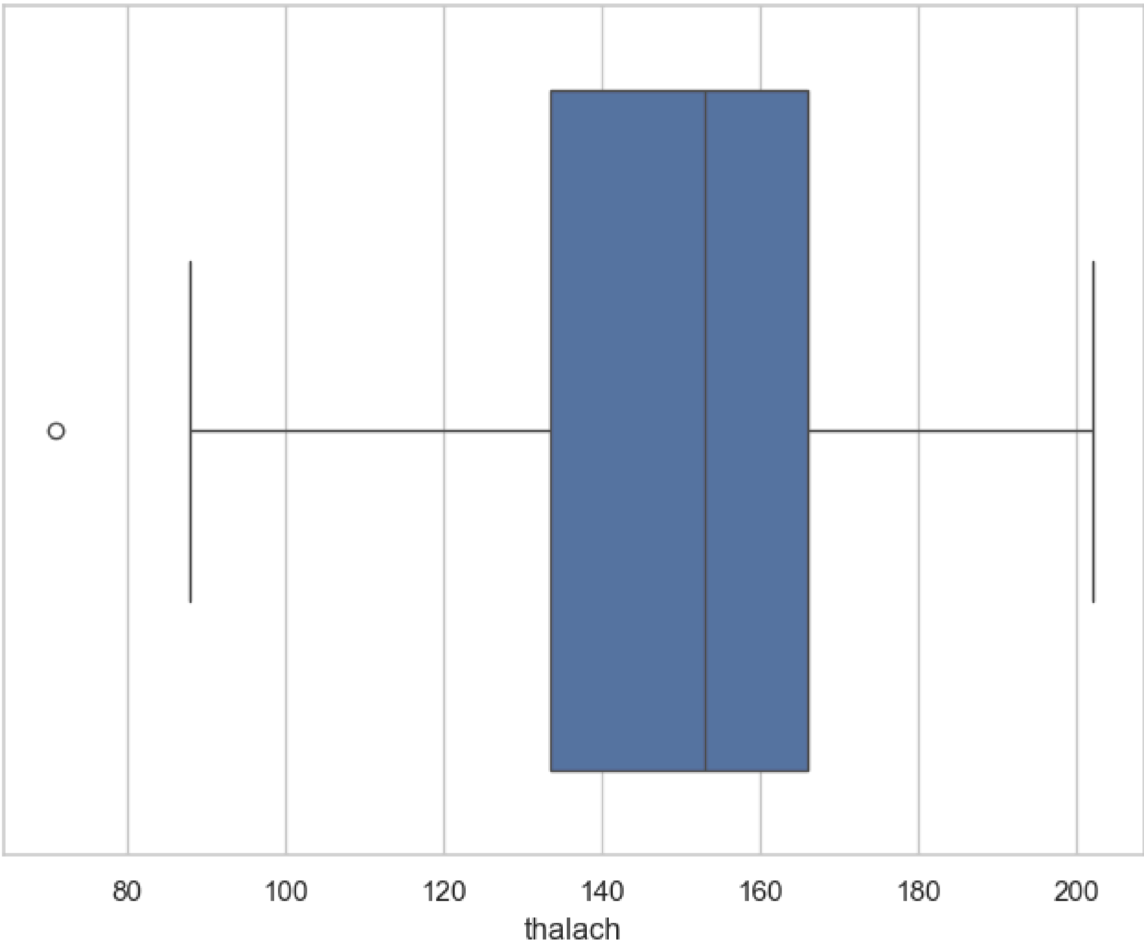
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["chol"])
plt.show()
```



In [226…
```
df['thalach'].describe()
```

Out[226…
```
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 [228…
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["thalach"])
plt.show()
```



In [230…

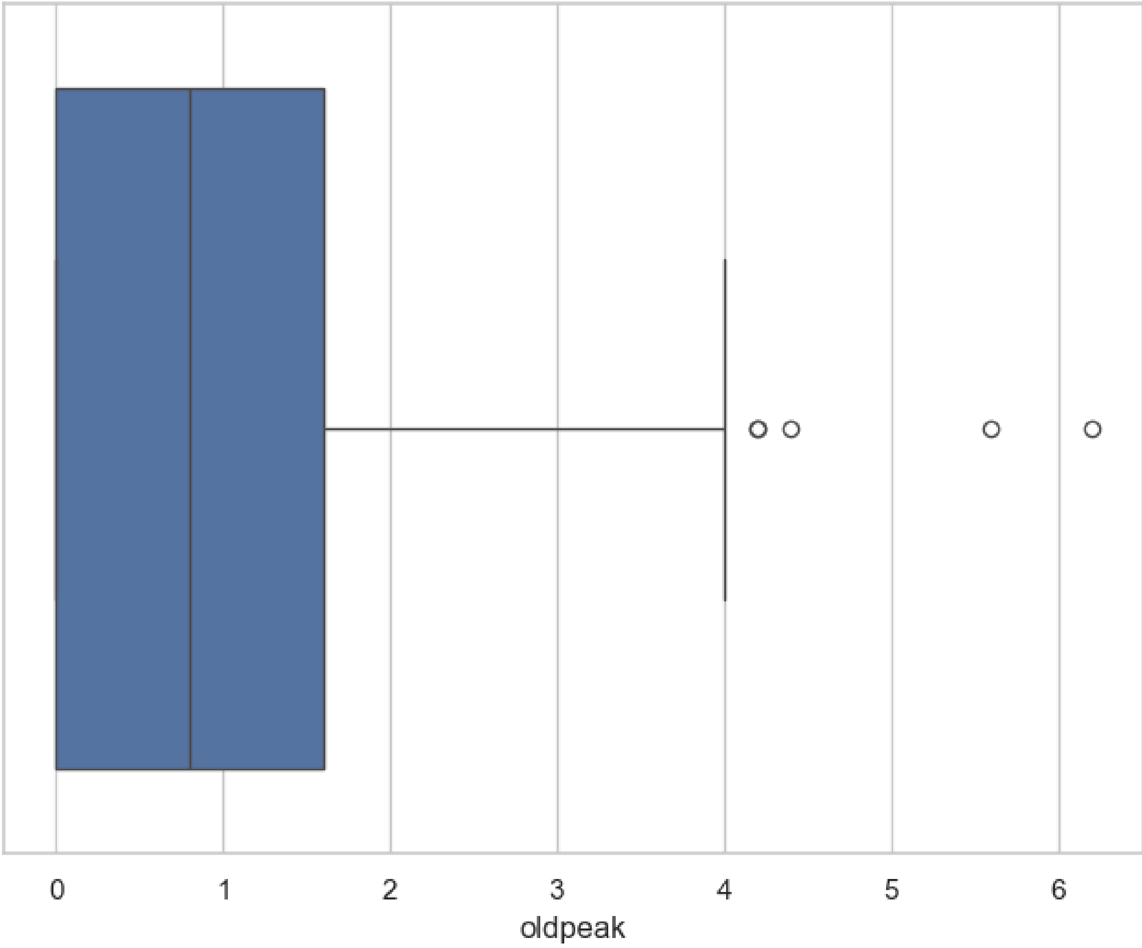
```
df['oldpeak'].describe()
```

Out[230…

```
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 [232…

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



Why EDA is Important:

*Identify Data Quality Issues: Helps in detecting errors, missing values, outliers, or inconsistencies in the dataset.

*Guide Further Analysis: By exploring the data first, you can make informed decisions about which statistical tests or machine learning algorithms to apply.

*Visual Insight: Visualization allows you to identify patterns, trends, and relationships that might not be obvious in raw data.

*Prevent Overfitting: It helps you understand the characteristics of the dataset, preventing you from overfitting your models.

EDA is not just about summarizing data but using those summaries to gain deeper insights, test assumptions, and prepare the data for further analysis or modeling.

Summary

In summary, EDA is both an art and a science. It involves iteratively exploring and visualizing data to better understand its structure and patterns, which helps in making informed decisions about further analysis, feature engineering, and modeling. The insights gained during EDA are foundational for any data science or statistical modeling task.

In [ ]: