heart-disease-analysis

October 29, 2024

1. Importing Libraries

[2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

2. Importing the CSV file.

[3]:
```
df = pd.read_csv(r"H:\DA. Python\5. Heart Disease Analysis\heart.csv")
```

3. Finding the shape of Data.

[4]:
```
df.shape
```

[4]:
```
(1025, 14)
```

4. Checking the first 5 rows of the data.

[5]:
```
df.head(5)
```

[5]:
```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   52    1   0       125   212    0        1      168      0      1.0      2
1   53    1   0       140   203    1        0      155      1      3.1      0
2   70    1   0       145   174    0        1      125      1      2.6      0
3   61    1   0       148   203    0        1      161      0      0.0      2
4   62    0   0       138   294    1        1      106      0      1.9      1

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

5. Checking the last 5 rows of the data.

[6]:
```
df.tail(5)
```

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

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

6. Getting Information About Our Dataset Like Total Number of Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

```
[7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

7. Check Null Values In The Dataset

```
[9]: df.isnull().sum()
```

```
[9]: 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
```

8. Check For Duplicate Data and Drop Them

[11]:
```
df_dup = df.duplicated().any()  #if the output is true, then it contains␣
 ↪duplicate values
print(df_dup)
```

```
True
```

[12]:
```
df = df.drop_duplicates()
```

[14]:
```
df.shape
```

[14]:
```
(302, 14)
```

9. Get Overall Statistics About The Dataset

[16]:
```
df.describe()
```

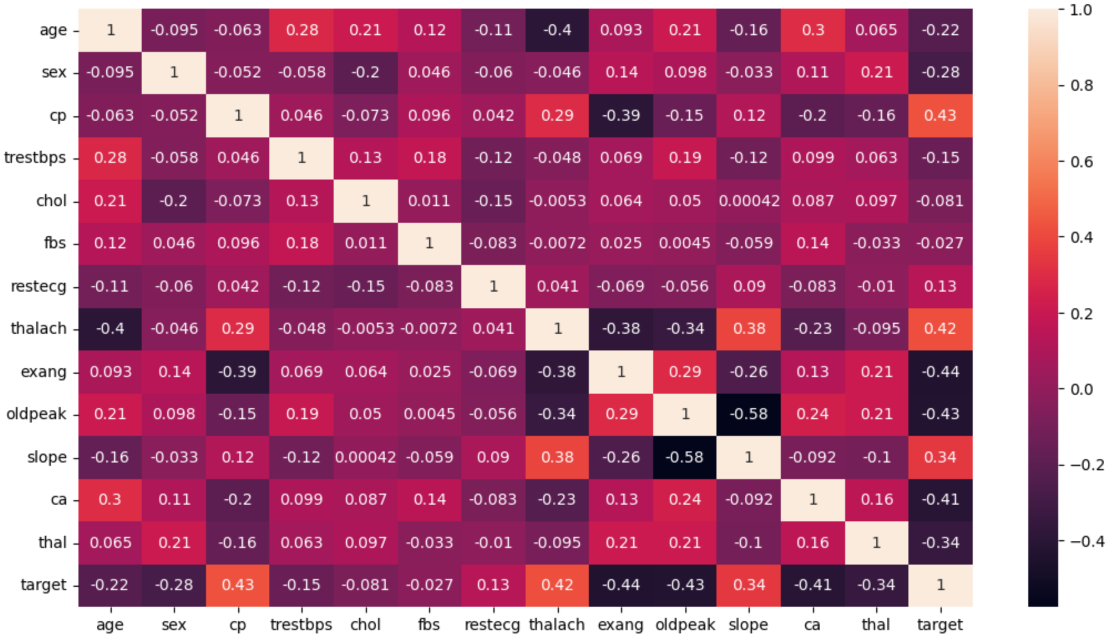
[16]:
```
              age         sex          cp    trestbps        chol         fbs  \
count  302.00000  302.000000  302.000000  302.000000  302.000000  302.000000
mean    54.42053    0.682119    0.963576  131.602649  246.500000    0.149007
std      9.04797    0.466426    1.032044   17.563394   51.753489    0.356686
min     29.00000    0.000000    0.000000   94.000000  126.000000    0.000000
25%     48.00000    0.000000    0.000000  120.000000  211.000000    0.000000
50%     55.50000    1.000000    1.000000  130.000000  240.500000    0.000000
75%     61.00000    1.000000    2.000000  140.000000  274.750000    0.000000
max     77.00000    1.000000    3.000000  200.000000  564.000000    1.000000

          restecg     thalach       exang     oldpeak       slope          ca  \
count  302.000000  302.000000  302.000000  302.000000  302.000000  302.000000
mean     0.526490  149.569536    0.327815    1.043046    1.397351    0.718543
std      0.526027   22.903527    0.470196    1.161452    0.616274    1.006748
min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000  133.250000    0.000000    0.000000    1.000000    0.000000
50%      1.000000  152.500000    0.000000    0.800000    1.000000    0.000000
```

```
75%      1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

             thal      target
count  302.000000  302.000000
mean     2.314570    0.543046
std      0.613026    0.498970
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

10. Creating a Correlation Matrix

[17]:
```
# to check the correlation between different features available in our dataset

plt.figure(figsize=(13,7))
sns.heatmap(df.corr(), annot=True)

# df.corr()
# sns.heatmap()
# plt.figure(figsize=(17,6))
# annot=True - parameter of this heatmap method of seaborn
```

[17]:
```
<Axes: >
```



11. How Many People Have Heart Disease, And How Many Don't Have Heart Disease In This Dataset?

```
[19]: df.columns
```

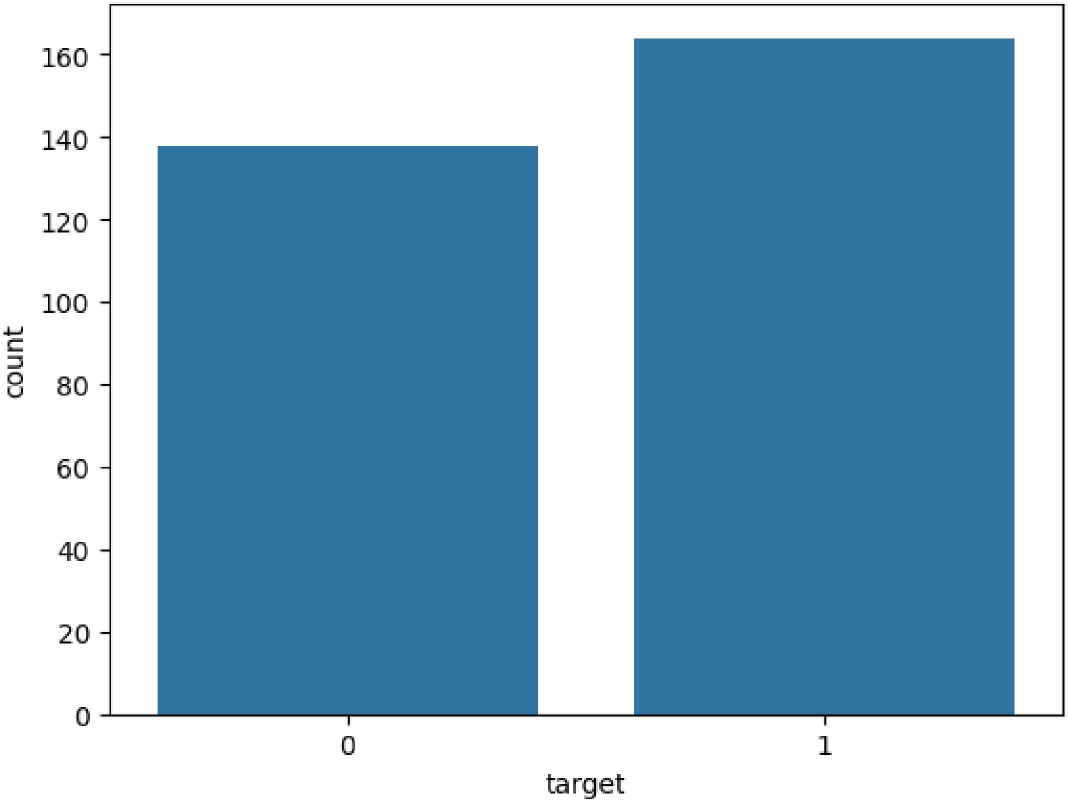
```
[19]: Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
             'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
            dtype='object')
```

```
[20]: df['target'].value_counts()

      # value_counts() - returns count of unique values in a column in descending␣
       ↪order
```

```
[20]: target
      1    164
      0    138
      Name: count, dtype: int64
```

```
[21]: sns.countplot(x= df['target'])

      # 1 - heart disease
```

```
[21]: <Axes: xlabel='target', ylabel='count'>
```



CONCLUSION = From this count plot it is clear that half of the people have heart disease

12. Find Count of Male & Female in this Dataset

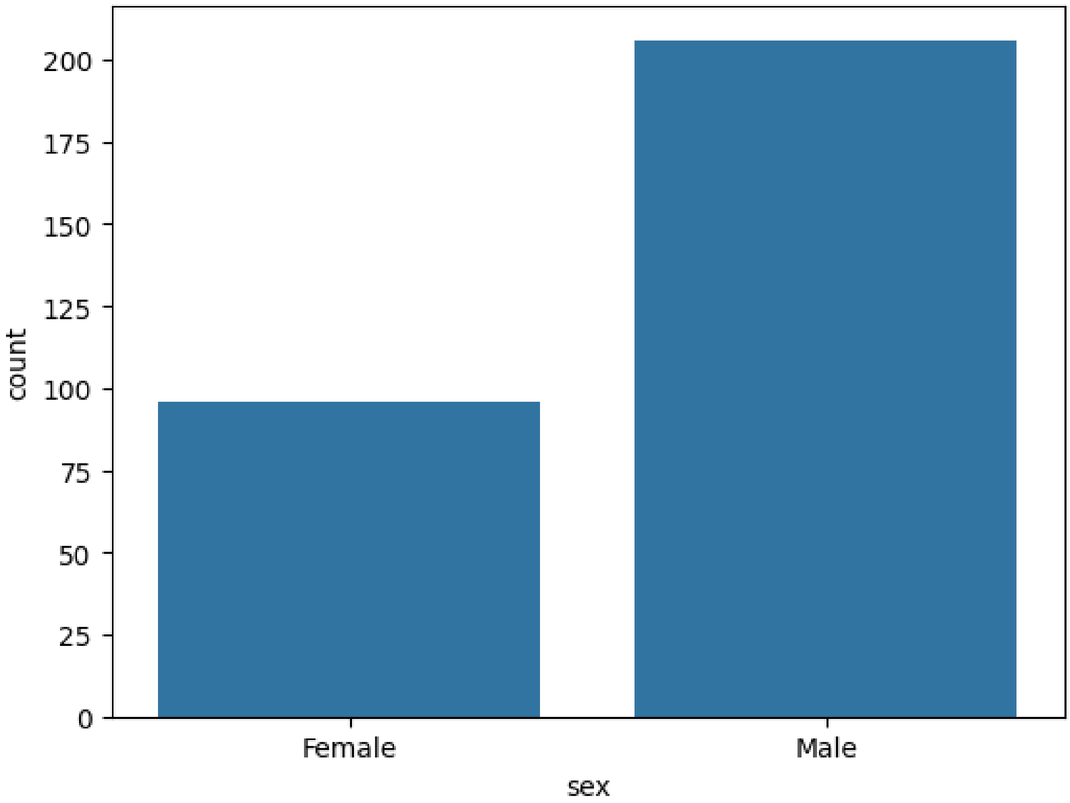
[25]:
```
df.columns
```

[25]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

[27]:
```
df['sex'].value_counts()
```

[27]:
```
sex
1    206
0     96
Name: count, dtype: int64
```

[31]:
```
# use countplot to visualize it

sns.countplot(x = df['sex'])

# change this x labels. [0,1] is replaced by ['Female','Male']

plt.xticks([0,1],['Female','Male'])
plt.show()
```



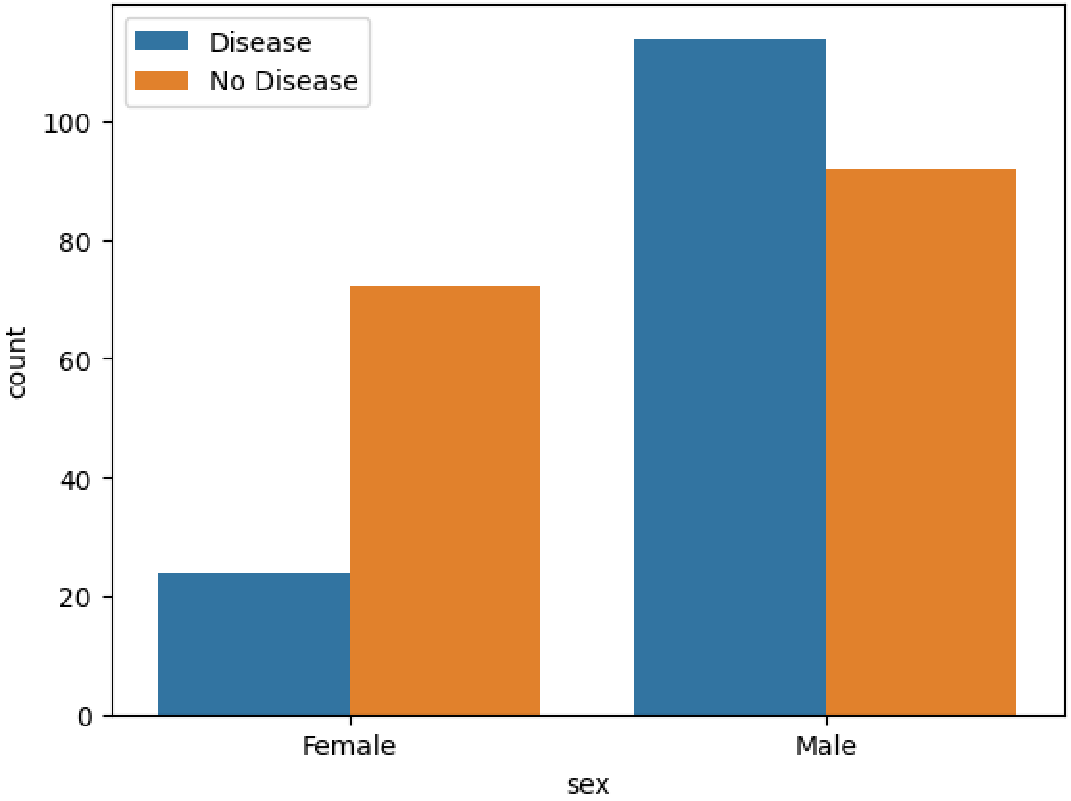
CONCLUSION = From this count plot it is clear that, approximately 30% of people are female and 70% are male.

13. Find Gender Distribution According to The Target Variable

[32]:
```
df.columns
```

[32]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

[41]:
```
#using countplot for distribution

sns.countplot(x='sex', hue = 'target', data = df)
plt.xticks([0,1],['Female','Male'])
plt.legend(labels=['Disease','No Disease'])
plt.show()
```

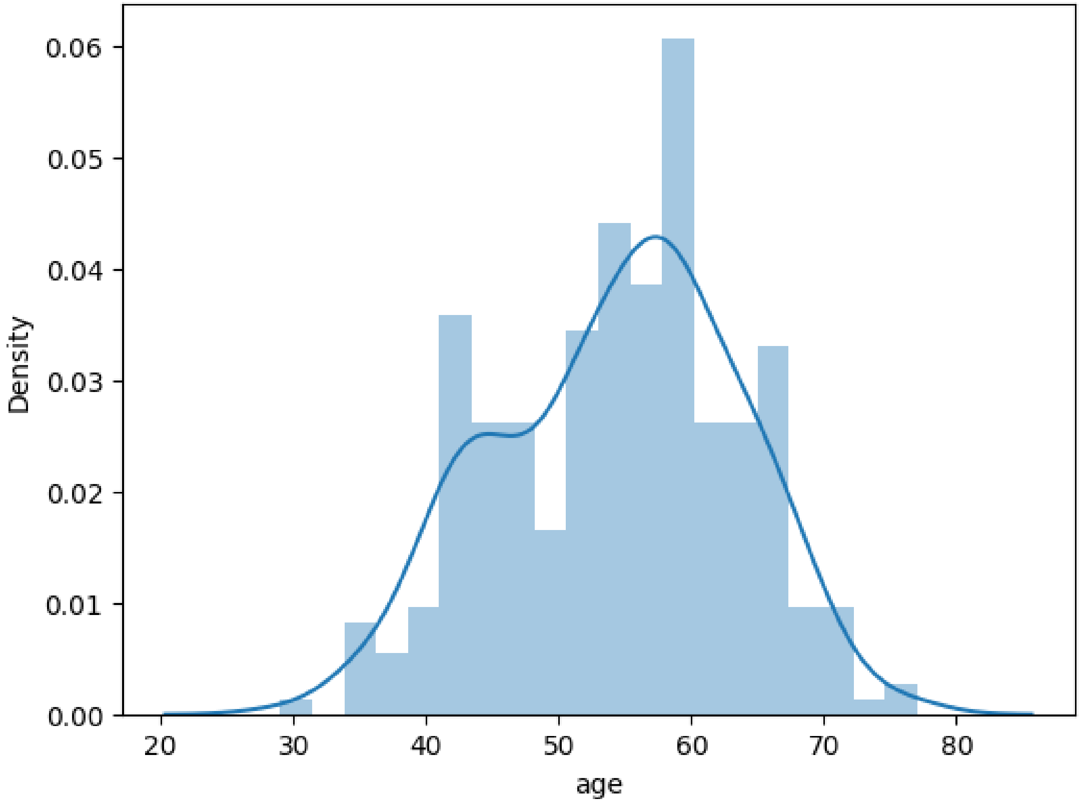


14. Check Age Distribution In The Dataset

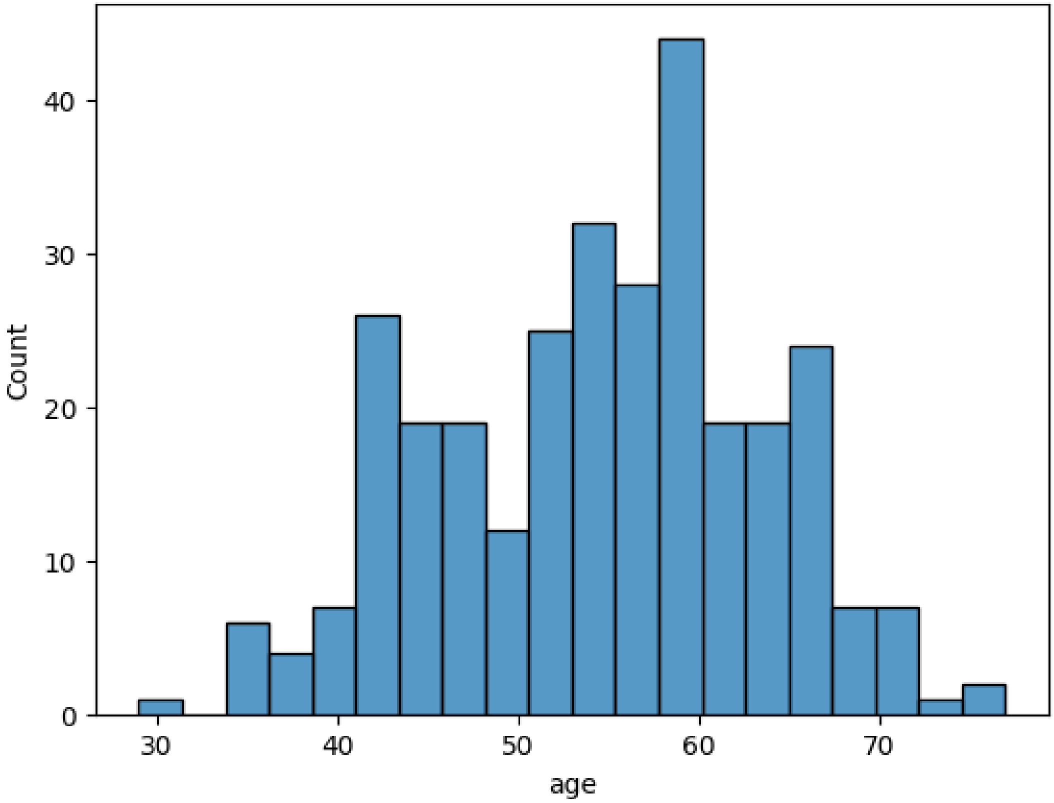
[43]:

```
sns.distplot(df['age'],bins=20)
plt.show()
```

```
C:\Users\Ebad\AppData\Local\Temp\ipykernel_1476\1602346454.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['age'],bins=20)
```



[44]:
```
sns.histplot(df['age'],bins=20)
plt.show()
```



CONCLUSION = From this plot we can see that most of the people in this dataset is aged between 50-60

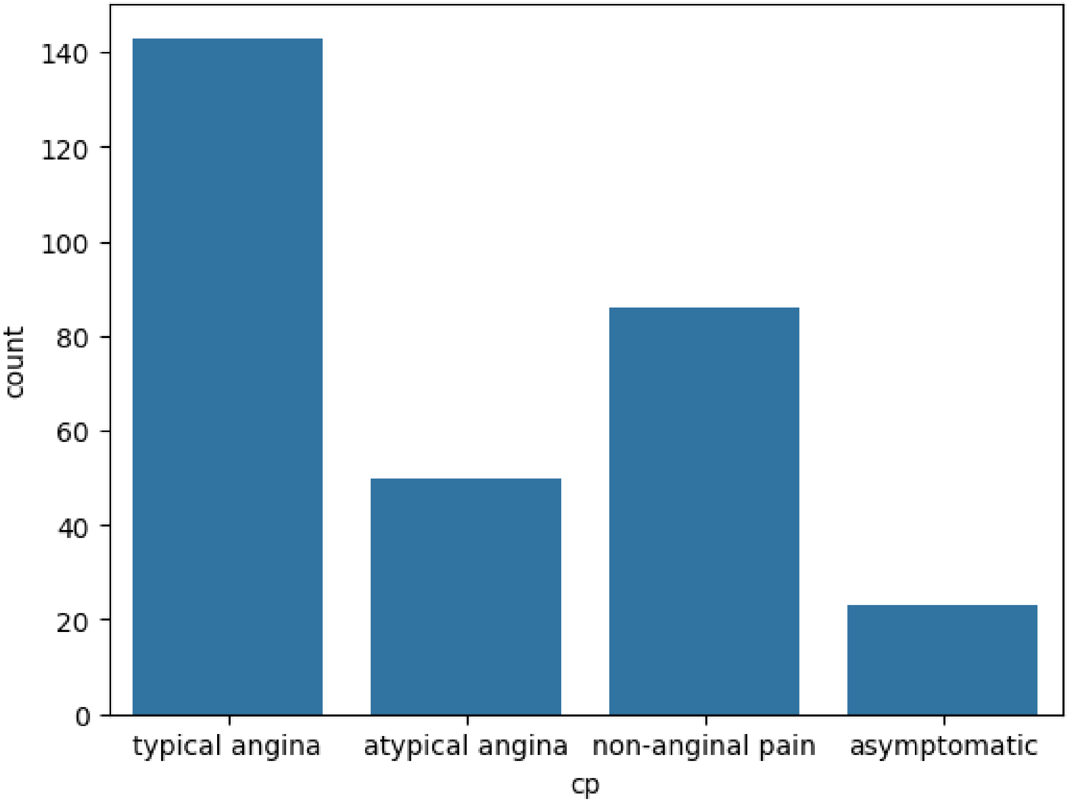
15. Checking Which Chest Pain Type is More Common

chest pain type (4 values):– value 0: typical angina– value 1: atypical angina– value 2: non-anginal pain– value 3: asymptomatic

[45]:
```
df.columns
```

[45]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

[54]:
```
sns.countplot(x=df['cp'])
plt.xticks([0,1,2,3],['typical angina','atypical angina','non-anginal␣
 ↪pain','asymptomatic'])
plt.xticks(rotation=0)
plt.show()
```



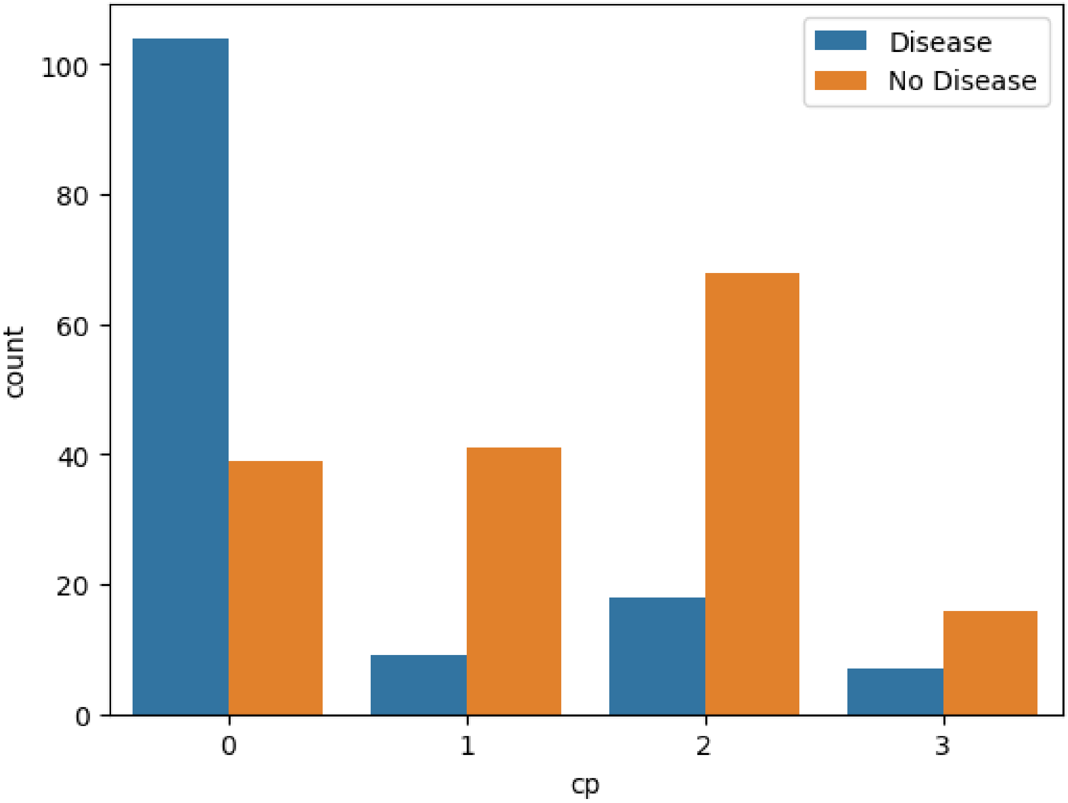
CONCLUSION = From this plot we can see that most common chest pain type is Typical Angima

16. Show The Chest Pain Distribution As Per Target Variable

[55]:
```
df.columns
```

[55]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

[64]:
```
sns.countplot(x ='cp', hue = 'target', data = df)
plt.xticks([0,1,2,3],['typical angina','atypical angina','non-anginal␣
 ↪pain','asymptomatic'])
plt.legend(['Disease','No Disease'])
plt.show()
```



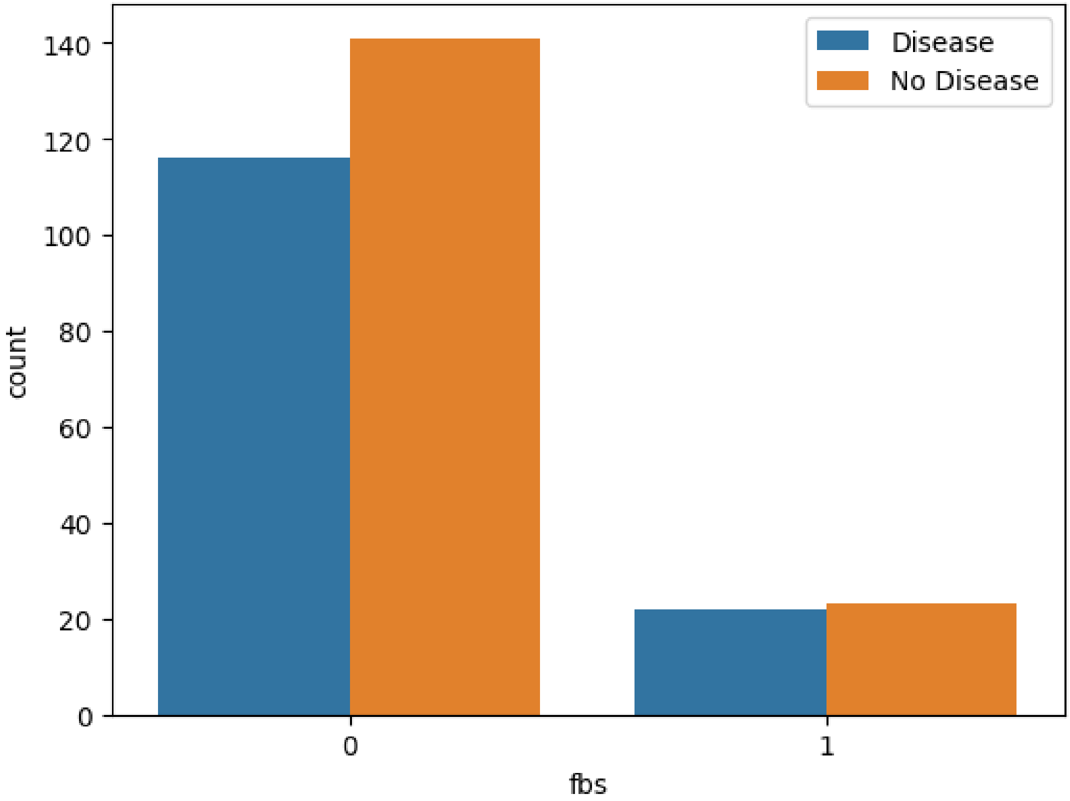
CONCLUSION = From this graph we can see that healthy people also have chest pain. Chest pain can be subjective. Due to stress, physical activities etc. It varies between gender.

17. Show Fasting Blood Sugar Distribution According To Target Variable

```
[68]: df.columns
```

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

```
[70]: sns.countplot(x='fbs', hue='target',data=df)
      plt.legend(labels=['Disease','No Disease'])
      plt.show()
```



CONCLUSION = FBS is a diabetic indicator. FBS greater than 120 are diabetics. Higher number of diabetics patient without heart disease.

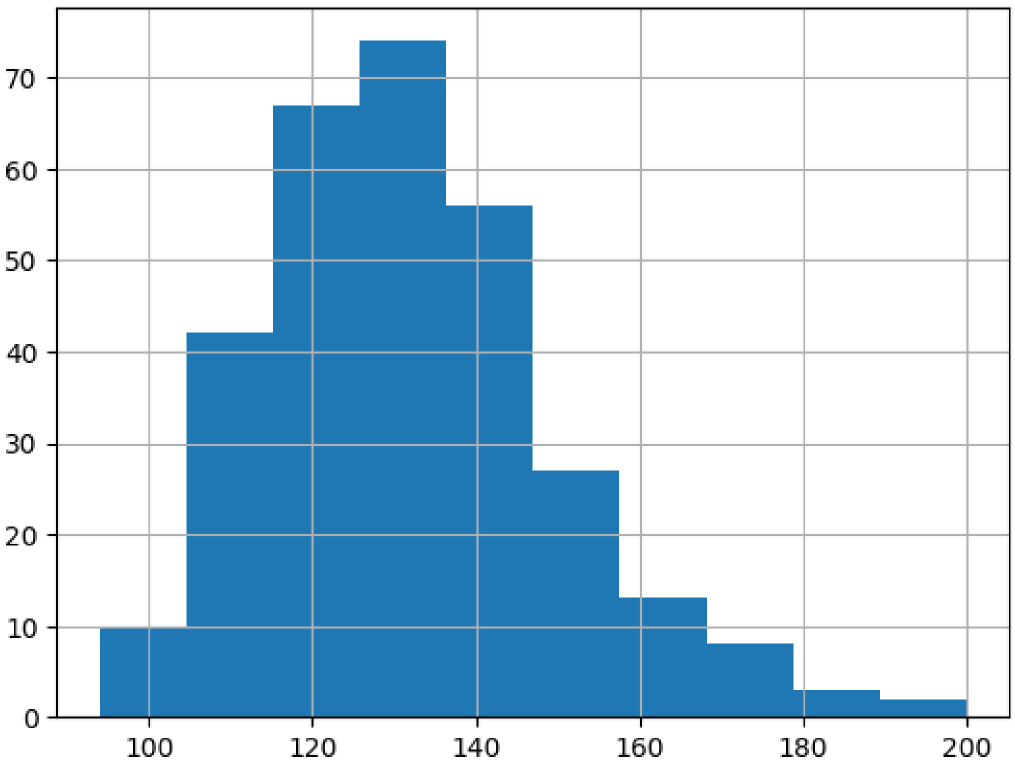
18. Check Resting Blood Pressure Distribution.

```
[71]: df.columns
```

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

```
[73]: df['trestbps'].hist()
```

```
[73]: <Axes: >
```



CONCLUSION = From this histogram we can see that the blood pressure of the people in this dataset is between 120 and 140

19. Compare Resting Blood Pressure As Per Sex Column

```
[75]: df.columns
```

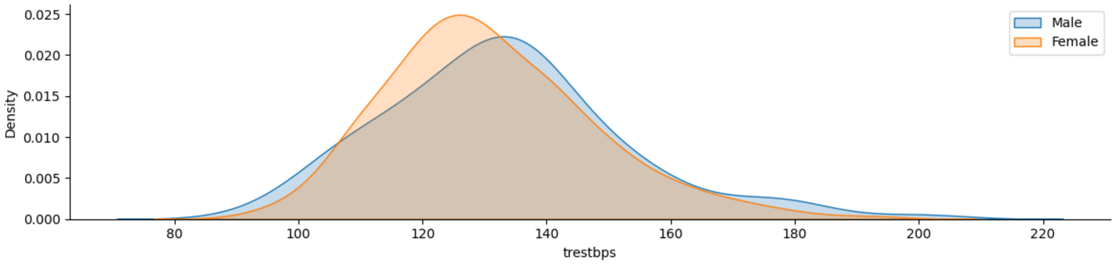
```
[75]: Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
             'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
            dtype='object')
```

```
[76]: # using "facetgrid class"

      g = sns.FacetGrid(df,hue="sex", aspect=4)
      g.map(sns.kdeplot,'trestbps',shade=True)
      plt.legend(labels=['Male','Female'])

      # we're using kdeplot of seaborn
      # we have to compare Resting BP as per sex column. so we have to pass "Resting␣
       ↪Blood Pressure" column. Here it is trestbps
```

```
c:\Users\Ebad\AppData\Local\Programs\Python\Python313\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:
```

```
`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
c:\Users\Ebad\AppData\Local\Programs\Python\Python313\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  func(*plot_args, **plot_kwargs)
```

```
[76]: <matplotlib.legend.Legend at 0x1ce273e4f50>
```



CONCLUSION = Woman have lower Resting blood pressure compared to men. For women is around 120, while for men it is little less than 140

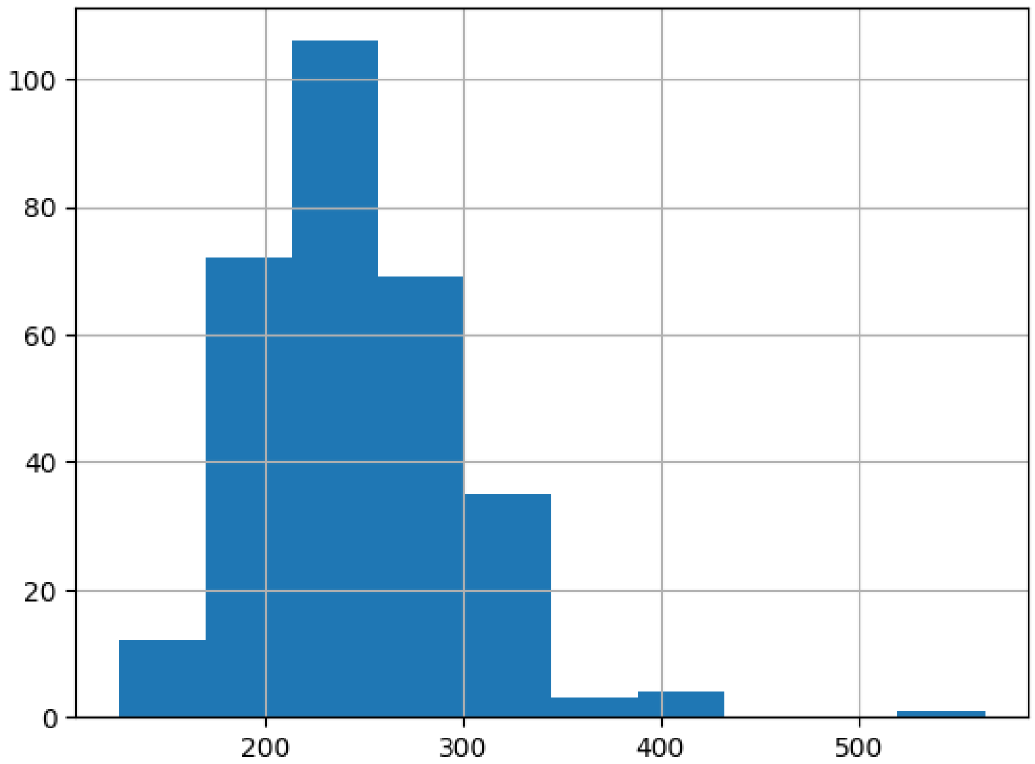
20. Show Distribution of Serum cholesterol

```
[77]: df.columns
```

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

```
[78]: df['chol'].hist()
```

```
[78]: <Axes: >
```



21. Plotting Continuous Variables

[79]:
```
# first we have to separate columns which contain continuous values and which contains categorical values.
```

[80]:
```
df.columns
```

[80]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

[82]:
```
# lets create 2 empty list.

categ_val=[]
cont_val=[]

for column in df.columns:
    if df[column].nunique() <=10:
        categ_val.append(column)
    else:
        cont_val.append(column)
```

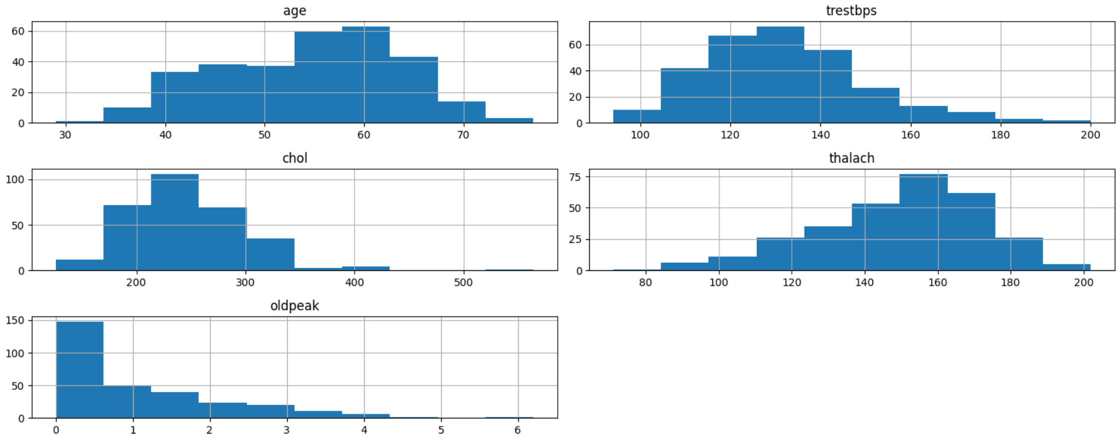
```
[83]: categ_val
```

```
[83]: ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
```

```
[84]: cont_val
```

```
[84]: ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

```
[86]: df.hist(cont_val,figsize=(15,6))
      plt.tight_layout()
      plt.show()
```


age
trestbps
chol
thalach
oldpeak