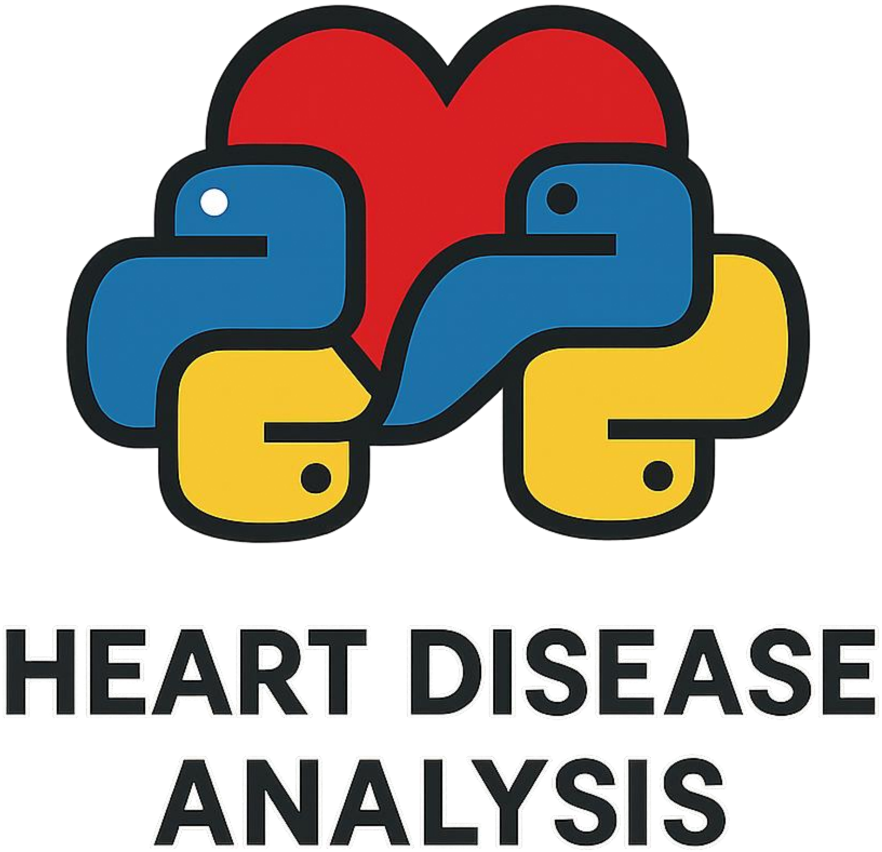
s



Context

This data set dates from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. It contains 76 attributes, including the predicted attribute, but all published experiments refer to using a subset of 14 of them. The "target" field refers to the presence of heart disease in the patient. It is integer valued 0 = no disease and 1 = disease.

Dataset Link :: https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

ss

Column Descriptions ::

- age
- sex
- chest pain type (4 values)

  - value 0: typical angina
  - value 1: atypical angina
  - value 2: non-anginal pain
  - value 3: asymptomatic
- trestbps: resting blood pressure (in mm Hg on admission to the hospital)
- chol: serum cholestrol in mg/dl
- fbs: (fasting blood sugar> 120 mg/dl)(1 = true; 0 = false)
- restecg: resting electrocardiographic results
  - value 0: normal
  - value 1: having ST-T wave abnormality(T wave inversions and/or ST elevation or depression of> 0.05 mV)
  - value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
- 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
  - value 1: upsloping
  - value 2: flat
  - value 3: downloping
- ca: number of major vessels (0-3) colored by flourosopy
- thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
- target: 0=less chance of heart attack, 1 = more chance of heart attack

This project covers manual exploratory data analysis and using pandas in Jupyter Notebook.

Questions:

1. Import The Libraries And Dataset
2. Display Top 5 Rows of The Dataset
3. Check The Last 5 Rows of The Dataset
4. Find Shape of Our Dataset (Number of Rows And Number of Columns)
5. Get Information About Our Dataset Like Total Number Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement
6. Check Null Values In The Dataset
7. Check For Duplicate Data and Drop Them
8. Get Overall Statistics About The Dataset
9. Draw Correlation Matrix
10. How Many People Have Heart Disease, And How Many Don't Have Heart Disease In This Dataset?
11. Find Count of Male & Female in this Dataset
12. Find Gender Distribution According to The Target Variable
13. Check Age Distribution In The Dataset
14. Check Chest Pain Type

15. Show The Chest Pain Distribution As Per Target Variable
16. Show Fasting Blood Sugar Distribution According To Target Variable
17. Check Resting Blood Pressure Distribution
18. Compare Resting Blood Pressure As Per Sex Column
19. Show Distribution of Serum cholesterol
20. Plot Continuous Variables

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

1. Import the libraries and dataset

In [2]:

```
df=pd.read_csv("heart.csv")
```

2. Display Top 5 Rows of The Dataset

In [3]:

```
df.head()
```

Out[3]:

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

3. Check the last 5 rows of the dataset

In [4]:

```
df.tail()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
1020	59	1	1	140	221	0	1	164	1	0.0	2	0
1021	60	1	0	125	258	0	0	141	1	2.8	1	1
1022	47	1	0	110	275	0	0	118	1	1.0	1	1
1023	50	0	0	110	254	0	0	159	0	0.0	2	0
1024	54	1	0	120	188	0	1	113	0	1.4	1	1

4. Find shape of our dataset (number of rows and number of columns)

```
In [5]: df.shape
```

```
Out[5]: (1025, 14)
```

```
In [6]: print("Number of rows: ", df.shape[0])
        print("Number of columns: ", df.shape[1])
```

```
Number of rows:  1025
Number of columns:  14
```

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

```
In [9]: 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
```

6. Check Null Values In The Dataset

```
In [8]: df.isnull().sum()
```

Out[8]:

```
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
```

7. Check For Duplicate Data and Drop Them

In [11]:

```
data_dup = df.duplicated().any()
print(data_dup)
```

```
True
```

In [12]:

```
df.duplicated().sum()
```

Out[12]:

```
np.int64(723)
```

In [13]:

```
df = df.drop_duplicates()
```

In [14]:

```
df.shape
```

Out[14]:

```
(302, 14)
```

8. Get Overall Statistics About The Dataset

In [15]:

```
df.describe()
```

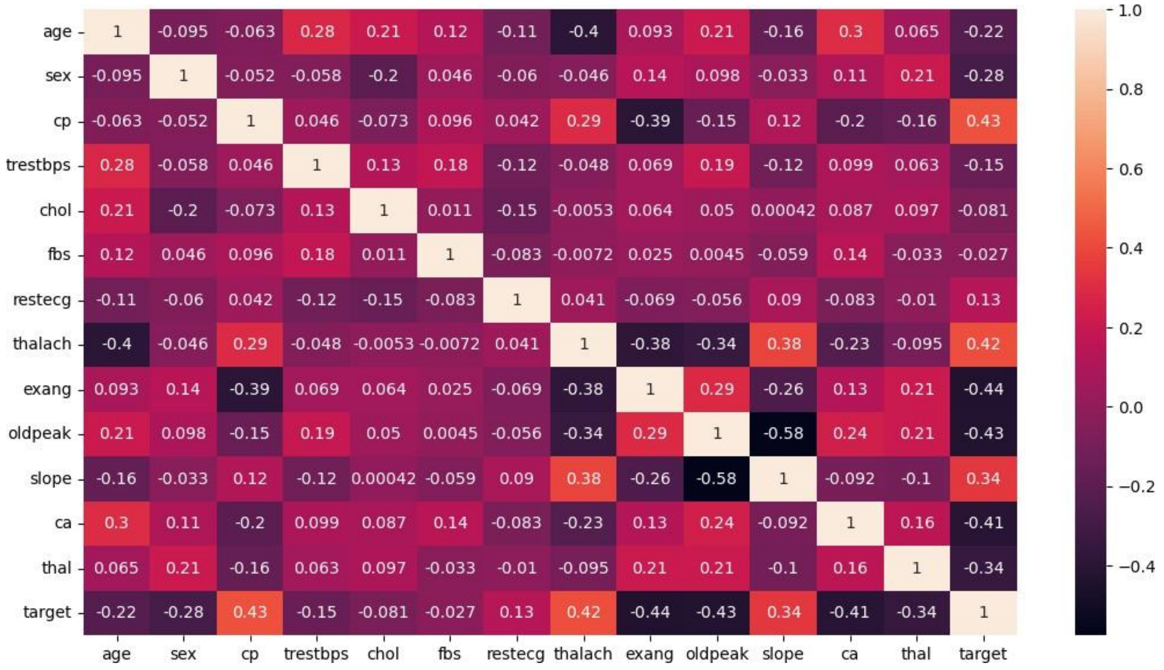
Out[15]:

	age	sex	cp	trestbps	chol	fbs	restec
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.0000
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.5264
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	0.5260
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	0.0000
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	0.0000
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	1.0000
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	1.0000
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	2.0000

9. Draw Correlation Matrix

In [16]:
```
# to check correlation between different features available in our dataset

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

# annot=True - parameter of this heatmap method of seaborn
```

Out[16]:
```
<Axes: >
```



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

In [17]:
```
df.columns
```

Out[17]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [18]:
```
df['target'].value_counts()
```

Out[18]:
```
target
1    164
0    138
Name: count, dtype: int64
```

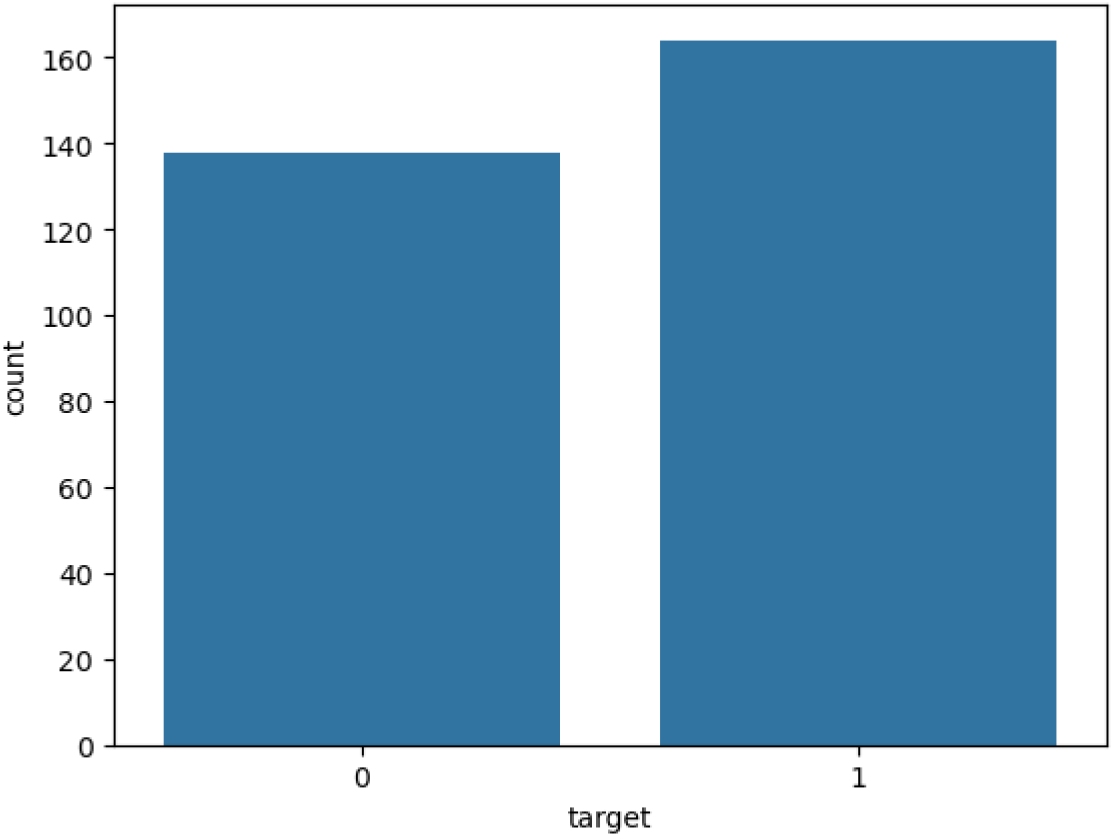
- 1 - heart disease
- 0 - NA

In [19]:
```
sns.countplot(x= df['target'])
```

```
# from this count plot it is clear that half of the people have heart disease
```

Out[19]:

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



11. Find Count of Male & Female in this Dataset

In [20]:

```
df.columns
```

Out[20]:

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

In [21]:

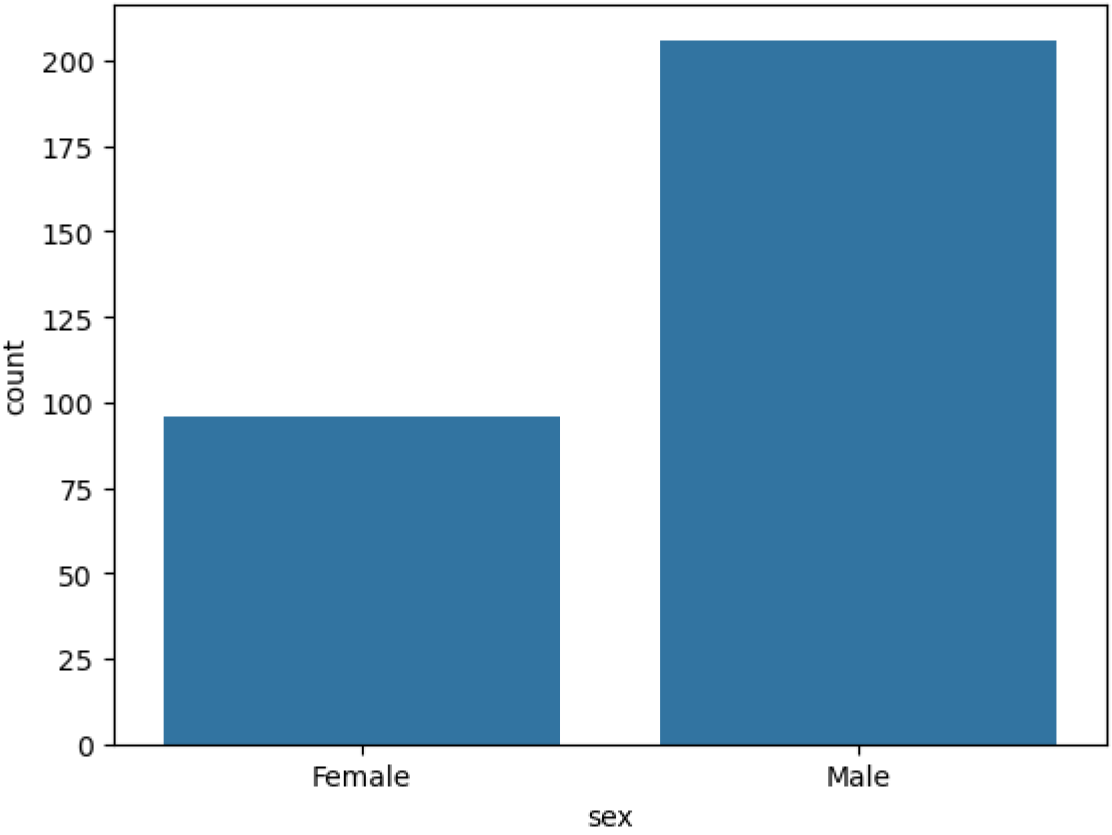
```
df['sex'].value_counts()
```

Out[21]:

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

In [22]:

```
# use countplot to visualize it

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

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

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

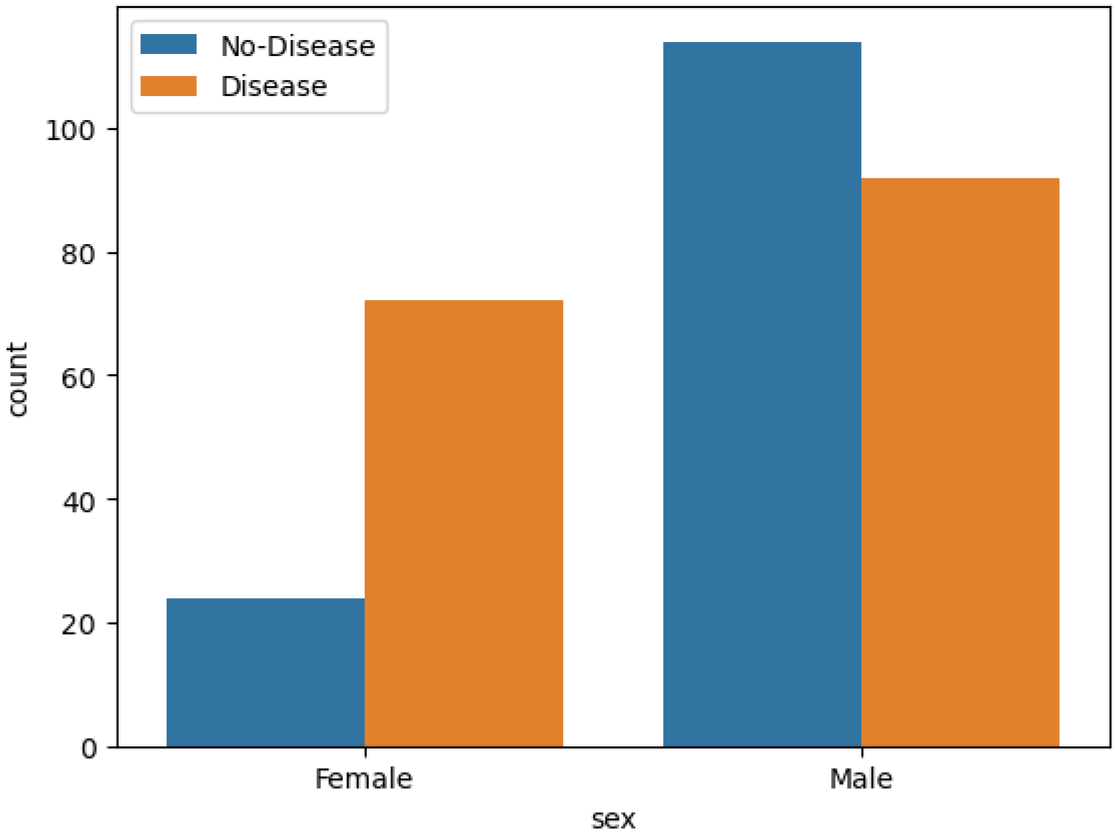


From the above count plot it is clear that, approximately 30% of people are female and 70% are male.

12. Find Gender Distribution According to The Target Variable.

In [23]:

```
# use "countplot" for distribution

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

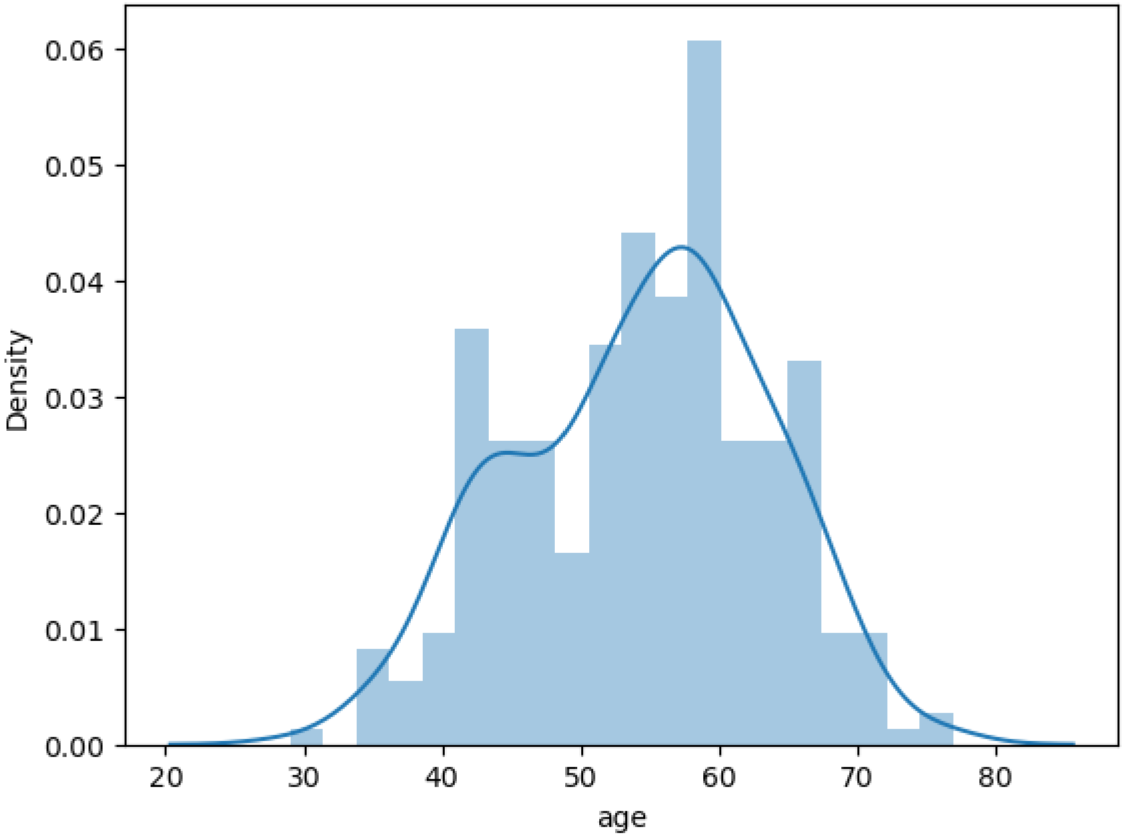


From this count plot, there are more men for disease and non-disease target.

13. Check Age Distribution In The Dataset

In [24]:
```
sns.distplot(df['age'],bins=20)
plt.show()
```

```
C:\Users\sanad\AppData\Local\Temp\ipykernel_23348\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)
```



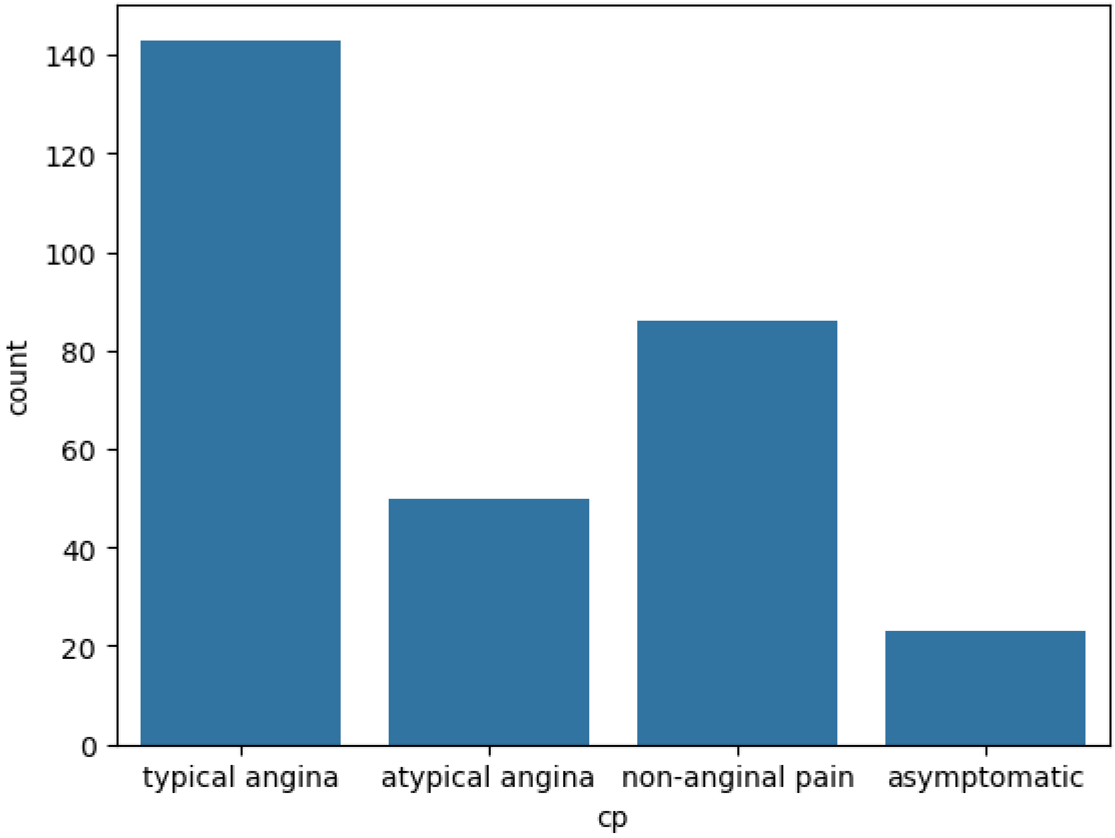
From this plot we can see that most of the people in this study aged between 50-60

14. Which Check 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

In [25]:

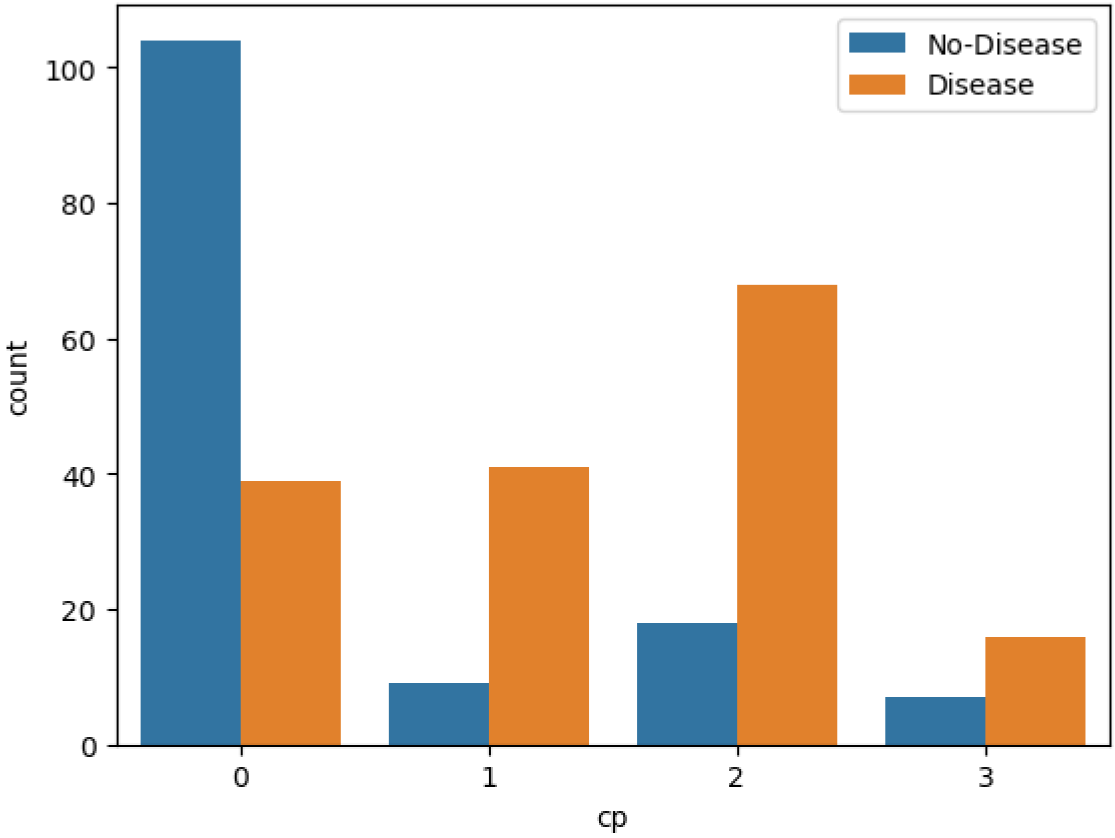
```
sns.countplot(x= df['cp'])
plt.xticks([0,1,2,3],["typical angina","atypical angina","non-anginal pain","asy
plt.xticks(rotation=0)
plt.show()
```



15. Show The Chest Pain Distribution As Per Target Variable

In [26]:

```
sns.countplot(x='cp',hue='target',  data=df)
plt.legend(labels=["No-Disease","Disease"])
plt.show()
```

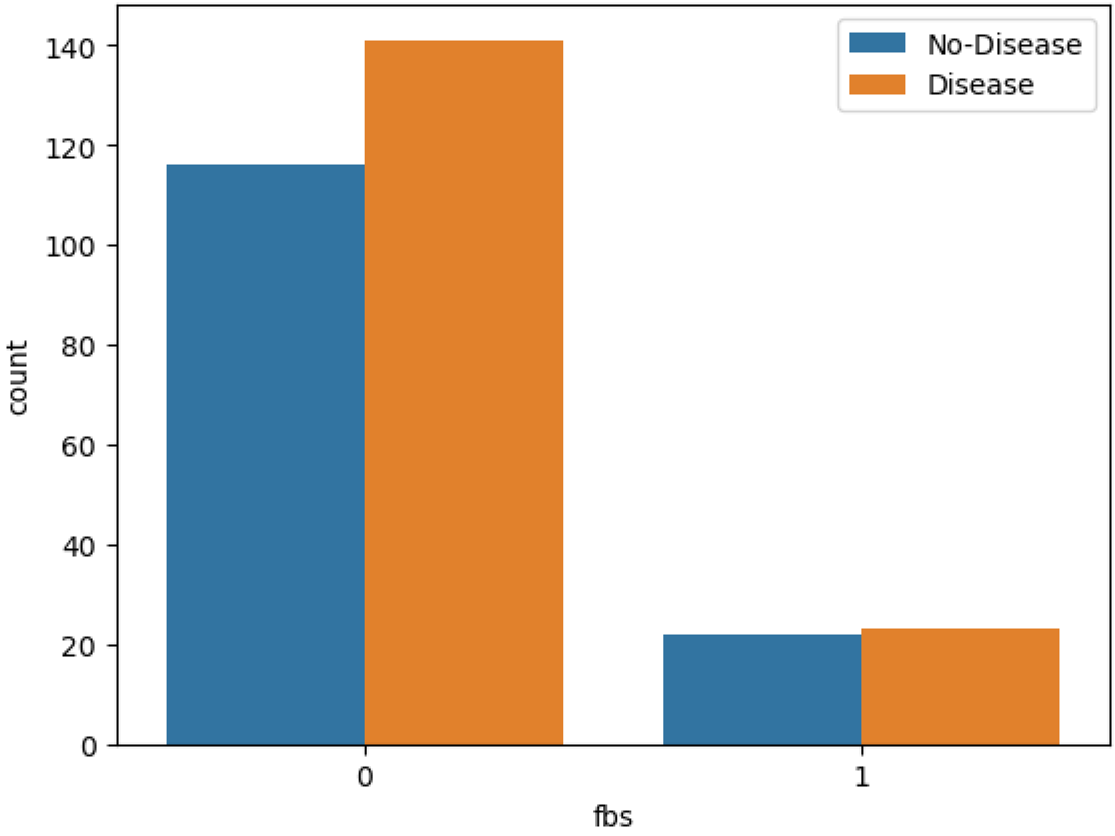


From this graph we can see that healthy people also have chest pain. Chest pain can be subjective. Due to stress, physical activities etc. It varies between gender

16. Show Fasting Blood Sugar Distribution According To Target Variable.

In [27]:

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

# fbs is a diabetic indicator
# fbs greater than 120 are diabetics
# higher number of diabetics patient without heart disease
```



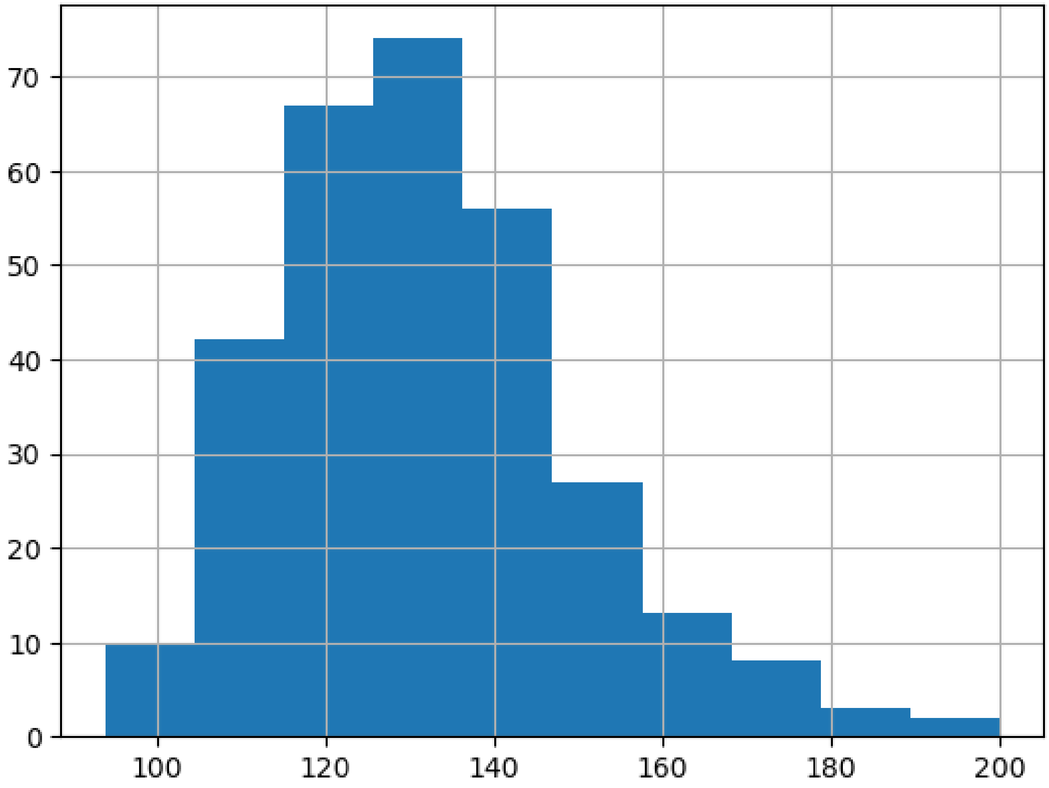
17. Check Resting Blood Pressure Distribution

In [28]:

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

Out[28]:

```
<Axes: >
```



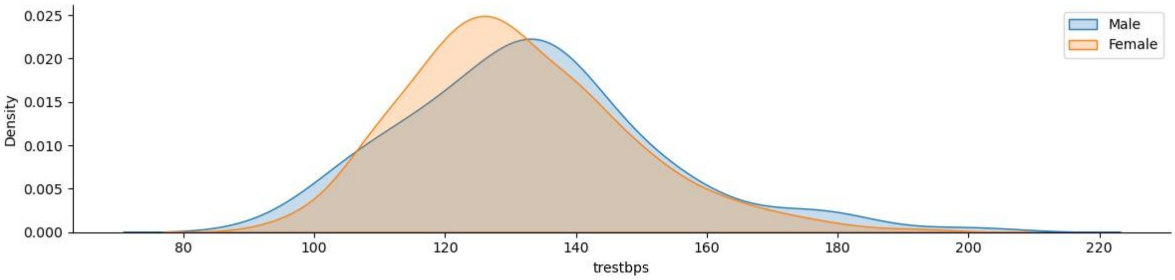
From this histogram we can see that the blood pressure of the people in this study is between 120 and 140

18. Compare Resting Blood Pressure As Per Sex Column

In [31]:
```
# lets use "facetgrade class"

"""facetgrade class is useful when you want to visualize the distribution of var
between multiple variables separately, within subset of your dataset."""

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 colu
so we have to pass "Resting Blood Pressure" column. Here it is trestbps"""
```

```
C:\Users\sanad\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: FutureWarnin
g:

`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\sanad\anaconda3\Lib\site-packages\seaborn\axisgrid.py:854: FutureWarnin
g:

`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)
```

Out[31]:
```
'we\'re using kdeplot of seaborn, we have to compare Resting BP as per sex colu
mn. \nso we have to pass "Resting Blood Pressure" column. Here it is trestbps'
```



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

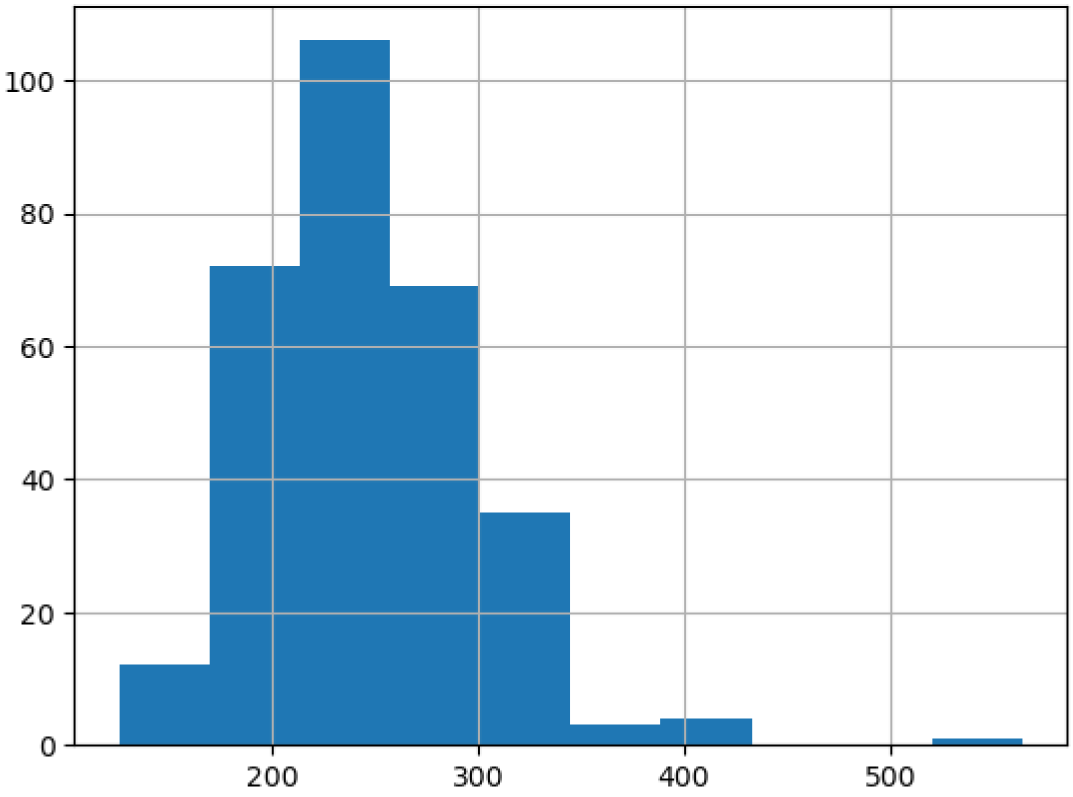
19. Show Distribution of Serum cholesterol

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

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

```
In [33]: # we are using histogram to check distribution of the column

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

```
Out[33]: <Axes: >
```



20. Plot Continuous Variables

In this question, we are gonna plot continuous variables.

```
In [34]: # first we have to separate columns which contain continuous values and which co

         df.columns
```

```
Out[34]: Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
                'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
               dtype='object')
```

```
In [35]: # lets create to 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)
```

```
In [36]: categ_val
```

```
Out[36]: ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
```

In [37]:

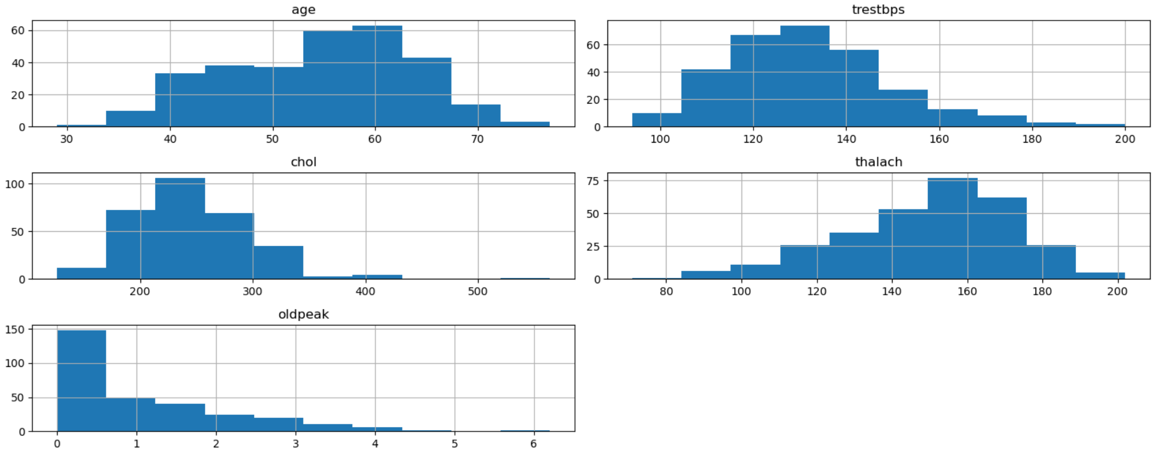
```
cont_val
```

Out[37]:

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

In [38]:

```
df.hist(cont_val,figsize=(15,6))
plt.tight_layout()
plt.show()
```



In [ ]: