In [1]:

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

In [28]:

```
url='heart.csv'
import pandas as pd
import numpy as np
df=pd.read_csv(url)
```

In [29]:

```
print(df.shape)
print(df.info())
```

```
(303, 14)
<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
None
```

Check data type The variables types are Binary: sex, fbs, exang, target Categorical: cp, restecg, slope, ca, thal Continuous: age, trestbps, chol, thalac, oldpeak

In [30]:

```
df.dtypes
```

Out[30]:

```
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 [31]:

```
# to know unique values
df.nunique()
```

Out[31]:

```
age          41
sex           2
cp            4
trestbps     49
chol        152
fbs           2
restecg       3
thalach      91
exang         2
oldpeak      40
slope         3
ca            5
thal          4
target        2
dtype: int64
```

In [32]:

```
# change the categorical type to categorical variables
df['sex'] = df['sex'].astype('object')
df['cp'] = df['cp'].astype('object')
df['fbs'] = df['fbs'].astype('object')
df['restecg'] = df['restecg'].astype('object')
df['exang'] = df['exang'].astype('object')
df['slope'] = df['slope'].astype('object')
df['ca'] = df['ca'].astype('object')
df['thal'] = df['thal'].astype('object')
df.dtypes
```

Out[32]:

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

Error Correction Check for the data characters mistakes feature ‘ca’ ranges from 0–3, however, df.nunique() listed 0–4. So lets find the ‘4’ and change them to NaN.

In [33]:

```
df['ca'].unique()
```

Out[33]:

```
array([0, 2, 1, 3, 4], dtype=object)
```

In [34]:

```
# to count the number in of each category decending order
print(df.ca.value_counts())

df[df['ca']==4]
```

```
0    175
1     65
2     38
3     20
4      5
Name: ca, dtype: int64
```

Out[34]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targ
92	52	1	2	138	223	0	1	169	0	0.0	2	4	2	
158	58	1	1	125	220	0	1	144	0	0.4	1	4	3	
163	38	1	2	138	175	0	1	173	0	0.0	2	4	2	
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	
251	43	1	0	132	247	1	0	143	1	0.1	1	4	3	

In [35]:

```
df.loc[df['ca']==4,'ca']=np.NaN
```

In [36]:

```
df['ca'].unique()
```

Out[36]:

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

Feature ‘thal’ ranges from 1–3, however, df.nunique() listed 0–3. There are two values of ‘0’. So lets change them to NaN

In [37]:

```
df.thal.value_counts()
```

Out[37]:

```
2    166
3    117
1     18
0      2
Name: thal, dtype: int64
```

In [38]:

```
df.loc[df['thal']==0,'thal']=np.NaN
```

In [39]:

```
df[df['thal']==0]
```

Out[39]:

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

In [40]:

```
df['thal'].unique()
```

Out[40]:

```
array([1, 2, 3, nan], dtype=object)
```

Check for missing values and replace **them**

In [41]:

```
df.isna().sum()
```

Out[41]:

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          5
thal        2
target      0
dtype: int64
```

In [42]:

```
df = df.fillna(df.median())
df.isnull().sum()
```

Out[42]:

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Check for duplicate rows

In [43]:

```
duplicated=df.duplicated().sum()
if duplicated:
  print("Duplicated rows :{}".format(duplicated))
else:
  print("No duplicates")
```

```
Duplicated rows :1
```

In [44]:

```
duplicates=df[df.duplicated(keep=False)]
duplicates.head()
```

Out[44]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
163	38	1	2	138	175	0	1	173	0	0.0	2	0.0	2.0	
164	38	1	2	138	175	0	1	173	0	0.0	2	0.0	2.0	

statistical summary check on the min and max value for the categorical variables (min-max). Sex (0–1), cp (0–3), fbs (0–1), restecg (0–2), exang (0–1), slope (0–2), ca (0–3), thal (0–3). Observe the mean, std, 25% and 75% on the continuous variables.

In [45]:

```
df.describe()
```

Out[45]:

	age	sex	cp	trestbps	chol	fbs	restecg	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	30
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	14
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	2
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	7
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	13
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	15
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	16
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	20

Before we plot the outliers, let's change the labeling for better visualization and interpretation.

In [46]:

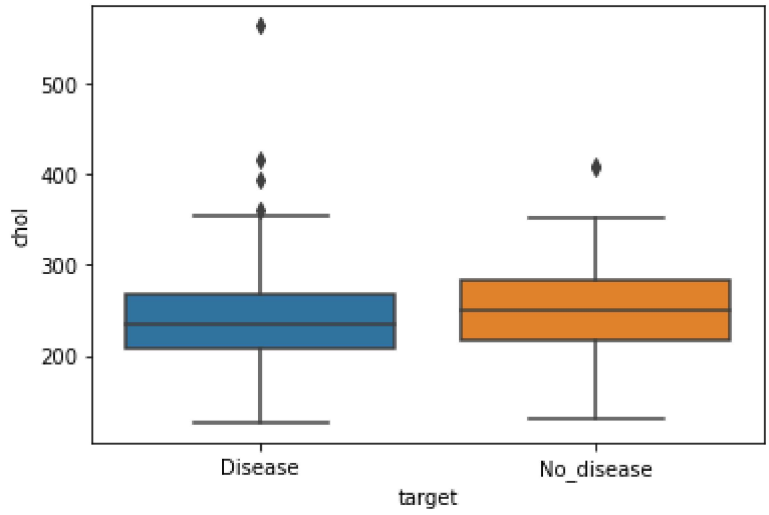
```
df['target'] = df.target.replace({1: "Disease", 0: "No_disease"})
df['sex'] = df.sex.replace({1: "Male", 0: "Female"})
df['cp'] = df.cp.replace({0: "typical_angina",
                          1: "atypical_angina",
                          2:"non-anginal pain",
                          3: "asymtomatic"})
df['exang'] = df.exang.replace({1: "Yes", 0: "No"})
df['fbs'] = df.fbs.replace({1: "True", 0: "False"})
df['slope'] = df.slope.replace({0: "upsloping", 1: "flat",2:"downsloping"})
df['thal'] = df.thal.replace({1: "fixed_defect", 2: "reversable_defect", 3:"normal"})
```

Outliers Detection & Handling

In [47]:

```
import matplotlib.pyplot as plt
import seaborn as sb
bxplt = sb.boxplot(df["target"],df["chol"])
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.
12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```

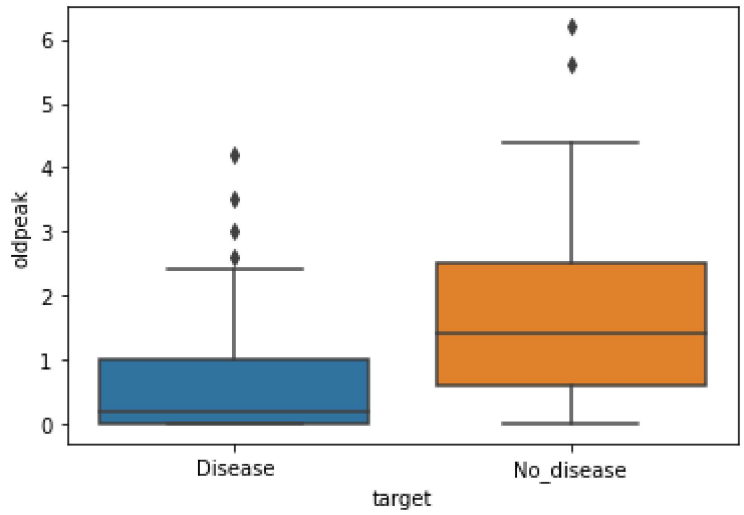


In [48]:

```
sb.boxplot(x='target', y='oldpeak', data=df)
```

Out[48]:

```
<AxesSubplot:xlabel='target', ylabel='oldpeak'>
```



In [49]:

```
# define continuous variable & plot
continous_features = ['age','trestbps','chol','thalach','oldpeak']
def outliers(df_out, drop = False):
    for each_feature in df_out.columns:
        feature_data = df_out[each_feature]
        Q1 = np.percentile(feature_data, 25.) # 25th percentile of the data of the given fe
        Q3 = np.percentile(feature_data, 75.) # 75th percentile of the data of the given fe
        IQR = Q3-Q1 #Interquartile Range
        outlier_step = IQR * 1.5 #That's we were talking about above
        outliers = feature_data[~((feature_data >= Q1 - outlier_step) & (feature_data <= Q3
        if not drop:
            print('For the feature {}, No of Outliers is {}'.format(each_feature, len(outli
        if drop:
            df.drop(outliers, inplace = True, errors = 'ignore')
            print('Outliers from {} feature removed'.format(each_feature))

outliers(df[continous_features])
```

```
For the feature age, No of Outliers is 0
For the feature trestbps, No of Outliers is 9
For the feature chol, No of Outliers is 5
For the feature thalach, No of Outliers is 1
For the feature oldpeak, No of Outliers is 5
```

Drop Outliers

In [50]:

```
outliers(df[continous_features],drop=True)
```

```
Outliers from age feature removed
Outliers from trestbps feature removed
Outliers from chol feature removed
Outliers from thalach feature removed
Outliers from oldpeak feature removed
```