In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#import missingno as msno # To visualize missing value
#import plotly.graph_objects as go # To Generate Graphs
#import plotly.express as px # To Generate box plot for statistical representat
%matplotlib inlineimport os
```

```
UsageError: unrecognized arguments: os
```

In [2]:
```
df=pd.read_csv("D:/DSBDAL Practical/heart.csv")
```

In [3]:
```
df.head(3)
```

Out[3]:

	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

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.2 KB
```

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

Out[6]:
```
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 [7]:
```
#Replace the NaN with median.
df = df.fillna(df.median())
```

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
```

In [9]:
```
# d) Check for duplicate rows
duplicates = df.duplicated(keep=False).sum()
duplicates
```

Out[9]:
```
2
```

In [10]:
```
duplicated=df[df.duplicated(keep=False)]
```

In [11]:
```
duplicated.head()
```

Out[11]:

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

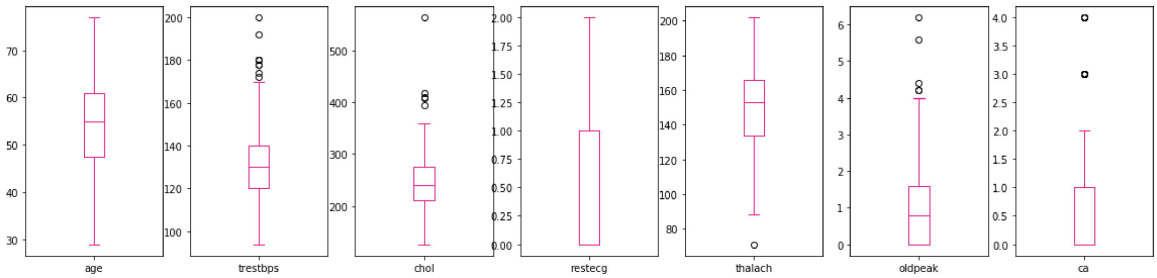
In [12]:
```
#e) Statistics summary
df.describe()
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202

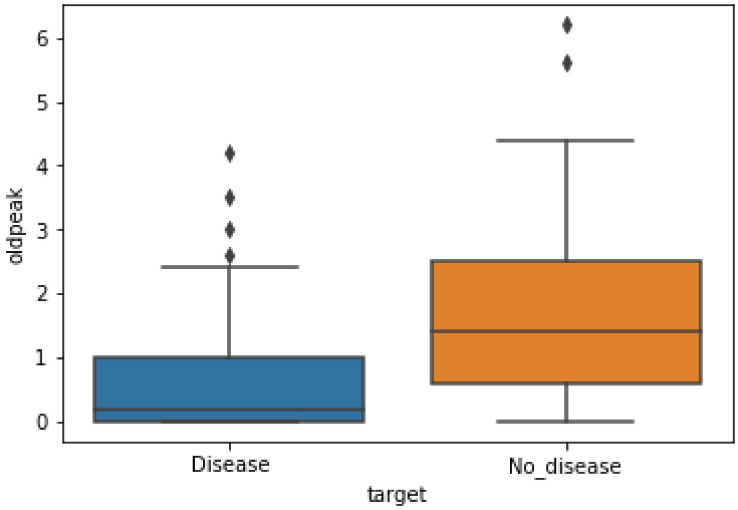
In [13]:
```
# Before we plot the outliers, let's change the labeling for better visualizati
df['target'] = df.target.replace({1: "Disease", 0: "No_disease"})
df['sex'] = df.sex.replace({1: "Male", 0: "Female"})
df['cp'] = df.cp.replace({0: "typical_angina",
                          1: "atypical_angina",
                          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:"nor
```

In [14]:
```
df.plot(kind='box', subplots=True, layout=(2,7),
sharex=False,sharey=False, figsize=(20, 10),
color='deeppink');
```



In [21]:
```
sns.boxplot(x='target', y='oldpeak', data=df)
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x1289d217710>



In [22]:
```
# 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
        Q3 = np.percentile(feature_data, 75.) # 75th percentile of the data of
        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) & (featur
        if not drop:
            print('For the feature {}, No of Outliers is {}'.format(each_featur
        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
```

In [23]:
```
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
```

In [24]:
```
print(df.target.value_counts())
```

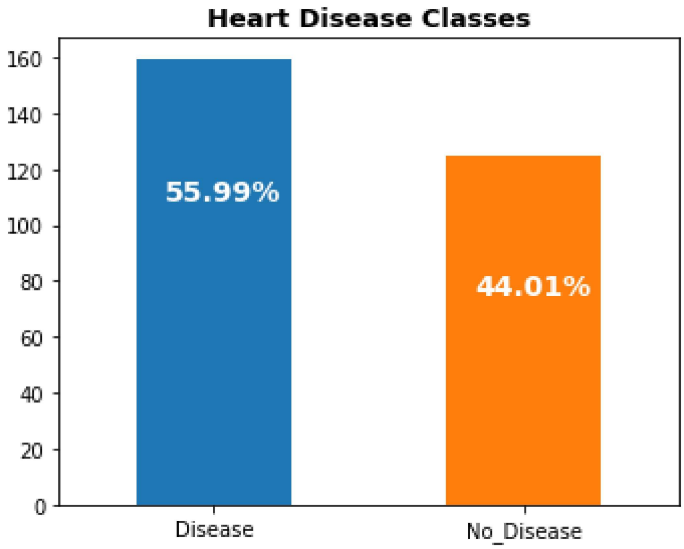
```
Disease       159
No_disease    125
Name: target, dtype: int64
```

In [25]:
```
# df['target'].value_counts().plot(kind='bar').set_title('Heart Disease Classes

fig, ax = plt.subplots(figsize=(5,4))
name = ["Disease", "No_Disease"]
ax = df.target.value_counts().plot(kind='bar')
ax.set_title("Heart Disease Classes", fontsize = 13, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)

# To calculate the percentage
totals = []
for i in ax.patches:
    totals.append(i.get_height())
total = sum(totals)
for i in ax.patches:
    ax.text(i.get_x()+.09, i.get_height()-50, \
            str(round((i.get_height()/total)*100, 2))+'%', fontsize=14,
                color='white', weight = 'bold')

plt.tight_layout()
```



In [ ]: