Importing the required Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
matplotlib.rcParams['figure.figsize']=(15,10)
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_excel('T:\Masters In Data Science\Machine Learning\Projects\Health_Care.xlsx')    ## Import the dataset
```

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

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
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [4]:
```
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 [5]:
```
df.isnull().sum()  ## Null values check
```

Out[5]:
```
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 [6]:
```
df.size
```

Out[6]:
```
4242
```

In [7]:
```
df.shape
```

Out[7]:
```
(303, 14)
```

In [8]:
```
df.drop_duplicates()  ## Drop the duplicate values if there are any this will help to avoid misleading results
```

Out[8]:

	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
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

302 rows × 14 columns

In [9]:
```
df.shape
```

Out[9]:
```
(303, 14)
```

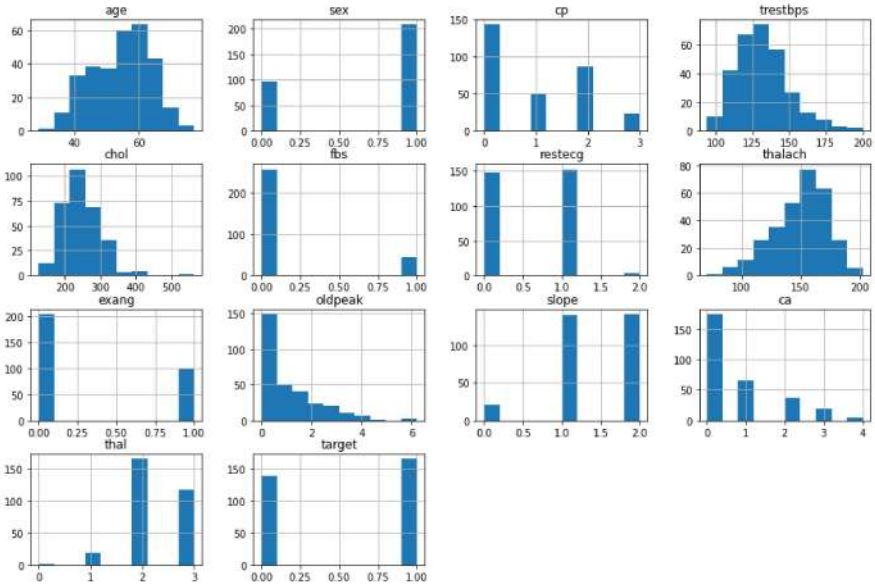
In [10]:
```
df.describe()
```

Out[10]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

In [11]:
```
df.hist()  ## Plot the Histogram to check the distribution of the data
```

Out[11]:
```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'sex'}>,
        <AxesSubplot:title={'center':'cp'}>,
        <AxesSubplot:title={'center':'trestbps'}>],
       [<AxesSubplot:title={'center':'chol'}>,
        <AxesSubplot:title={'center':'fbs'}>,
        <AxesSubplot:title={'center':'restecg'}>,
        <AxesSubplot:title={'center':'thalach'}>],
       [<AxesSubplot:title={'center':'exang'}>,
        <AxesSubplot:title={'center':'oldpeak'}>,
        <AxesSubplot:title={'center':'slope'}>,
        <AxesSubplot:title={'center':'ca'}>],
       [<AxesSubplot:title={'center':'thal'}>,
        <AxesSubplot:title={'center':'target'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



Looks like chol colun has outliers we will remove them in the further steps

In [14]:
```
def classs (x):              ## This function will divide data with age-wise groups
    if 0<x<=25:
        return 'Below_25'
    elif 26<x<=35:
        return '25_to_35'
    elif 36<x<=45:
        return '35_to_45'
    elif 46<x<=55:
        return '45_to_55'
    elif 56<x<=65:
        return '55_to_65'
    else:
        return 'Above_65'
```

In [13]:
```
df['Class'] = df['age'].apply(classs)
```

In [15]:
```
df.head()
```

Out[15]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target	Class
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1	55_to_65
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1	35_to_45
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1	35_to_45
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1	Above_65
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1	55_to_65

In [16]:
```
age = df.groupby(['target','Class']).size().reset_index().rename(columns={0:'Count'})
```

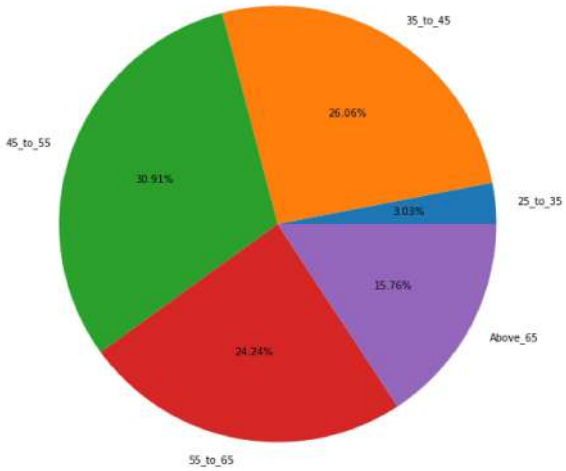
In [17]:
```
age
```

Out[17]:

	target	Class	Count
0	0	25_to_35	2
1	0	35_to_45	14
2	0	45_to_55	30
3	0	55_to_65	67
4	0	Above_65	25
5	1	25_to_35	5
6	1	35_to_45	43
7	1	45_to_55	51
8	1	55_to_65	40
9	1	Above_65	26

In [18]:
```
plt.pie(age['Count'][5:],labels = age['Class'][5:],autopct='%1.2f%%')
```

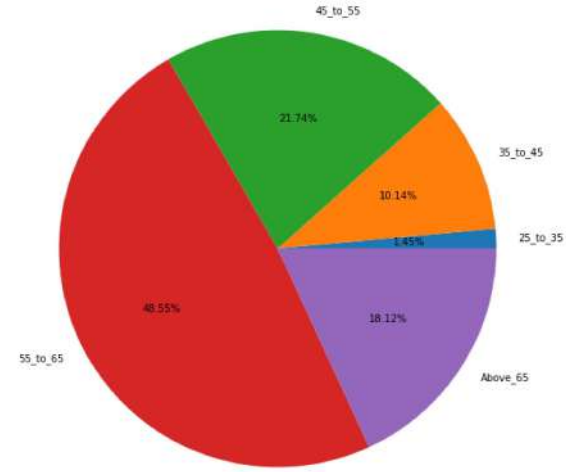
Out[18]:
```
([<matplotlib.patches.Wedge at 0x2156a2ed310>,
  <matplotlib.patches.Wedge at 0x2156a2eda30>,
  <matplotlib.patches.Wedge at 0x2156a2fb190>,
  <matplotlib.patches.Wedge at 0x2156a2fb8b0>,
  <matplotlib.patches.Wedge at 0x2156a2fbfd0>],
 [Text(1.0950191145337336, 0.10456165074136119, '25_to_35'),
  Text(0.5858685213332059, 0.9309984294889241, '35_to_45'),
  Text(-1.0360287823840757, 0.3696543819187994, '45_to_55'),
  Text(-0.19788437336962075, -1.0820544232043565, '55_to_65'),
  Text(0.967945285667399, -0.5225724102497735, 'Above_65')],
 [Text(0.5972831533820364, 0.0570336276771061, '3.03%'),
  Text(0.3195646479993045, 0.5078173251757767, '26.06%'),
  Text(-0.5651066085731321, 0.20162966286479964, '30.91%'),
  Text(-0.10793693092888404, -0.5902115035660126, '24.24%'),
  Text(0.5279701558185812, -0.2850394964998764, '15.76%')])
```



Pie chart clearly shows that out of all people having CVD 30.96% people are from 45 to 55 age group

In [19]:
```
plt.pie(age['Count'][:5],labels = age['Class'][:5],autopct='%1.2f%%')
```

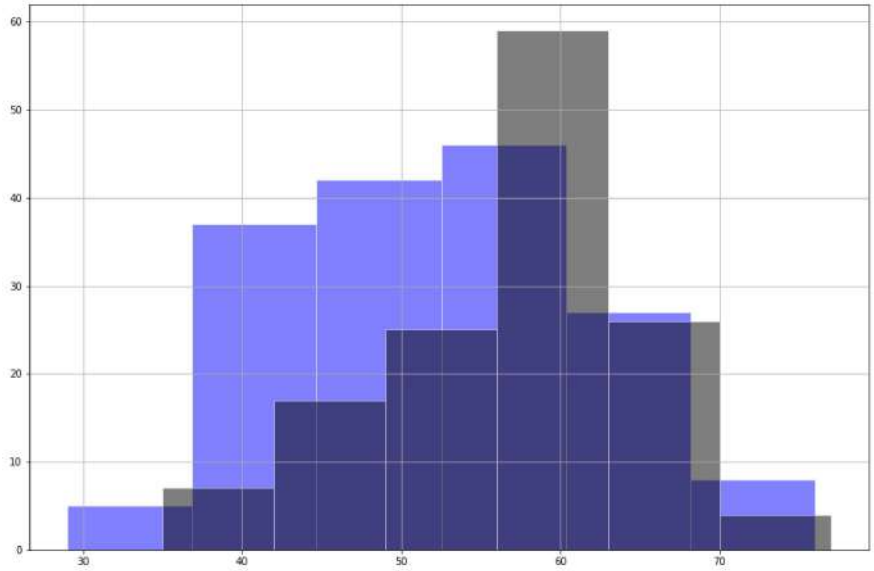
Out[19]:
```
([<matplotlib.patches.Wedge at 0x2156a488a00>,
  <matplotlib.patches.Wedge at 0x2156a497160>,
  <matplotlib.patches.Wedge at 0x2156a497880>,
  <matplotlib.patches.Wedge at 0x2156a497fa0>,
  <matplotlib.patches.Wedge at 0x2156a4a3700>],
 [Text(1.098860040968941, 0.0500660599781685, '25_to_35'),
  Text(1.0089324334399594, 0.43824119472377554, '35_to_45'),
  Text(0.17455083610198344, 1.0860626158818372, '45_to_55'),
  Text(-0.9766737623884708, -0.5060714987646994, '55_to_65'),
  Text(0.9266076645058751, -0.5927885255965807, 'Above_65')],
 [Text(0.5993782041648769, 0.027308759988091905, '1.45%'),
  Text(0.5503267818763414, 0.2390406516675139, '10.14%'),
  Text(0.09520954696471823, 0.5923977904810019, '21.74%'),
  Text(-0.532731143120984, -0.27603899932619963, '48.55%'),
  Text(0.5054223624577 5, -0.32333919577995307, '18.12%')])
```



Pie chart shows that out of people not having CVD 48.55% people are from 55 to 65 age group

In [20]:
```
df[df['target']==1]['age'].hist(alpha = 0.5, color ='b',bins=6,label = 'target = 1',edgecolor='White')
df[df['target']==0]['age'].hist(alpha = 0.5, color ='k',bins=6,label = 'target = 0',edgecolor='White')
```

Out[20]:
```
<AxesSubplot:>
```



In [21]:
```
sex = df.groupby(['sex','Class']).size().reset_index().rename(columns={0:'Count'})
```

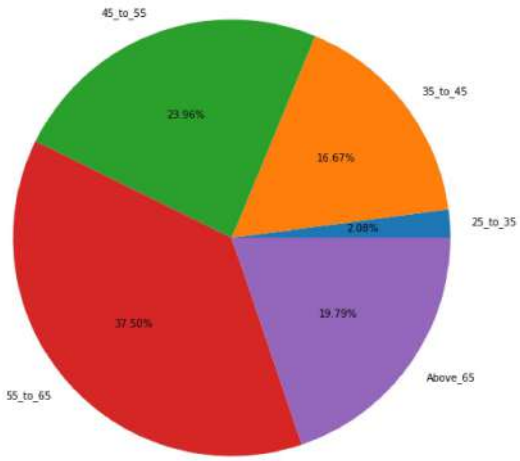
In [22]:
```
sex
```

Out[22]:

	sex	Class	Count
0	0	25_to_35	2
1	0	35_to_45	16
2	0	45_to_55	23
3	0	55_to_65	36
4	0	Above_65	19
5	1	25_to_35	5
6	1	35_to_45	41
7	1	45_to_55	58
8	1	55_to_65	71
9	1	Above_65	32

In [23]:
```
plt.pie(sex['Count'][0:5],labels = age['Class'][0:5],autopct='%1.2f%%')
```

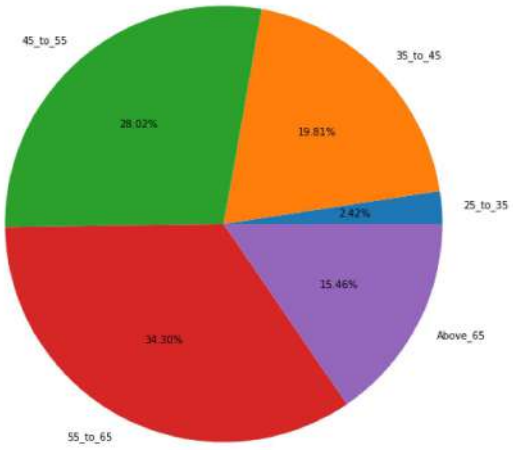
Out[23]:
```
([<matplotlib.patches.Wedge at 0x2156a56c0d0>,
  <matplotlib.patches.Wedge at 0x2156a56c7f0>,
  <matplotlib.patches.Wedge at 0x2156a56cf10>,
  <matplotlib.patches.Wedge at 0x2156a579610>,
  <matplotlib.patches.Wedge at 0x2156a579d30>],
 [Text(1.097644815422134, 0.07194344429417662, '25_to_35'),
  Text(0.8726886612586932, 0.6696375889318864, '35_to_45'),
  Text(-0.3874750727942776, 1.0294965118751347, '45_to_55'),
  Text(-0.827023785397473, -0.7252803998363906, '55_to_65'),
  Text(0.8941313655484946, -0.6407254491140372, 'Above_65')],
 [Text(0.5987153538666186, 0.03924187870591452, '2.08%'),
  Text(0.4760119970501962, 0.36525686669011986, '16.67%'),
  Text(-0.21135003970596958, 0.5615435519318916, '23.96%'),
  Text(-0.4511038829440761, -0.3956074908198494, '37.50%'),
  Text(0.48770801757190607, -0.34948660860765657, '19.79%')])
```



Pie chart shows that out of all females 37.50% females are from age group of 55 to 65 years

In [24]:
```
plt.pie(sex['Count'][5:],labels = age['Class'][5:],autopct='%1.2f%%')
```

Out[24]:
```
([<matplotlib.patches.Wedge at 0x2156a9f1760>,
  <matplotlib.patches.Wedge at 0x2156a9f1ee0>,
  <matplotlib.patches.Wedge at 0x2156aa01640>,
  <matplotlib.patches.Wedge at 0x2156aa01d60>,
  <matplotlib.patches.Wedge at 0x2156aa0e4c0>],
 [Text(1.0968344198640751, 0.08339217829891288, '25_to_35'),
  Text(0.7866204494036967, 0.7689136938434159, '35_to_45'),
  Text(-0.7134403143142207, 0.8372591700968263, '45_to_55'),
  Text(-0.506071521625399, -0.9766737505430119, '55_to_65'),
  Text(0.9728053931507576, -0.5134682726876122, 'Above_65')],
 [Text(0.5982733199258591, 0.04548664270849793, '2.42%'),
  Text(0.4290656996747436, 0.4194074693691359, '19.81%'),
  Text(-0.38914926235321123, 0.45668682005281425, '28.02%'),
  Text(-0.27603901179567214, -0.5327311366598246, '34.30%'),
  Text(0.5306211235367768, -0.2800736032841521, '15.46%')])
```



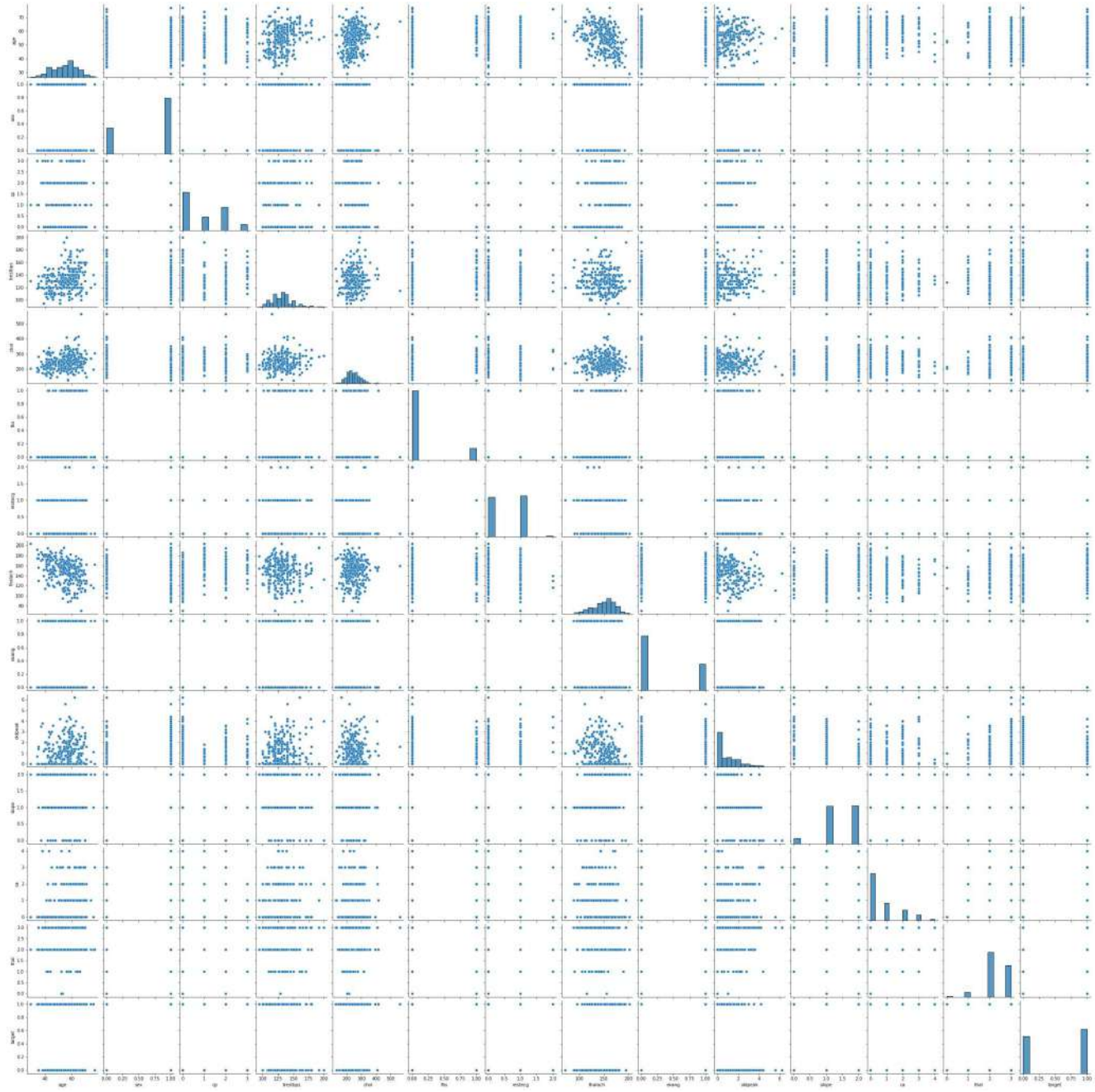
Pie chart shows that out of all males 34.30% males are from age group of 55 to 65 years

In [25]:
```
sns.heatmap(df.corr(), annot=True)  ## Checking the correlation between the variables
```

Out[25]:
```
<AxesSubplot:>
```

In [26]:
```
sns.pairplot(df)
```

Out[26]:
```
<seaborn.axisgrid.PairGrid at 0x2156a31d0d0>
```



In [27]:
```
df.columns
```

Out[27]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target', 'Class'],
      dtype='object')
```

In [28]:
```
df[['trestbps','target']].corr()
```

Out[28]:

	trestbps	target
trestbps	1.000000	-0.144931
target	-0.144931	1.000000

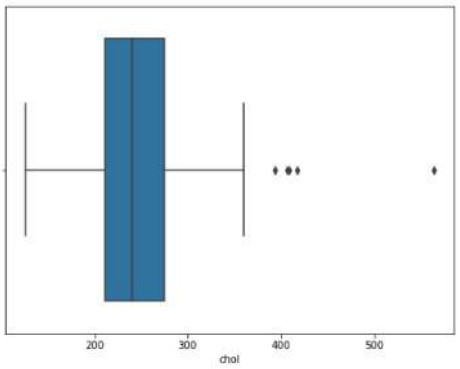
In [29]:
```
df.head()
```

Out[29]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target	Class
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1	55_to_65
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1	35_to_45
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1	35_to_45
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1	Above_65
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1	55_to_65

In [30]:
```
plt.figure(figsize=(8,6))
sns.boxplot(data=df,x=df['chol'])
```

Out[30]:
```
<AxesSubplot:xlabel='chol'>
```



There are outliers in the column

In [31]:
```
Q1 = df['chol'].quantile(0.25)
Q3 = df['chol'].quantile(0.75)
```

In [32]:
```
IQR = Q3-Q1
IQR
```

Out[32]: 63.5

In [33]:
```
lower_whisker = Q1-1.5*IQR
upper_whisker = Q3+1.5*IQR
```

In [34]:
```
lower_whisker
```

Out[34]: 115.75

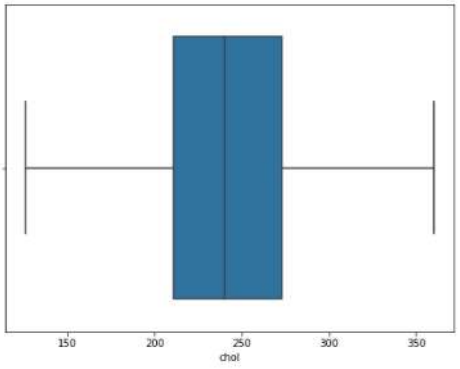
In [35]:
```
upper_whisker
```

Out[35]: 369.75

In [36]:
```
new_df = df[df['chol']<upper_whisker]
```

In [37]:
```
plt.figure(figsize=(8,6))
sns.boxplot(data=new_df,x=new_df['chol'])
```

Out[37]: <AxesSubplot:xlabel='chol'>



Let us check by using chi2 test if Blood Pressure of person can detect heart attack

H0 = Anomalies in trestbps can detect heart attack

H1 = Anomalies in trestbps can not detect heart attack

In [38]:
```
new_df_crosstab_trestbps = pd.crosstab(new_df['trestbps'],new_df['target'],margins=True)
new_df_crosstab_trestbps
```

Out[38]:

target	0	1	All
trestbps			
94	0	2	2
100	2	2	4
101	0	1	1
102	0	2	2
104	0	1	1
105	0	3	3
106	0	1	1
108	2	4	6
110	11	8	19
112	4	5	9
114	1	0	1
115	0	2	2
117	1	0	1
118	2	5	7
120	14	23	37
122	1	3	4
123	1	0	1
124	4	2	6
125	7	4	11
126	2	1	3
128	6	6	12
129	0	1	1
130	13	23	36
132	5	3	8
134	2	2	4
135	1	5	6
136	2	1	3
138	3	10	13
140	15	15	30
142	1	2	3
144	2	0	2
145	4	1	5
146	1	1	2
148	1	1	2
150	7	9	16
152	3	2	5
154	1	0	1
155	0	1	1
156	0	1	1
160	6	5	11
164	1	0	1
165	1	0	1
170	3	1	4
172	0	1	1
174	1	0	1
178	1	1	2
180	2	1	3
192	1	0	1
200	1	0	1
All	136	162	298

In [39]:
```
from scipy.stats import chi2_contingency
chi2_contingency(new_df_crosstab_trestbps)
```

Out[39]:
```
(46.68606210694453,
 0.999997573052193,
 98,
 array([[  0.91275168,   1.08724832,   2.        ],
        [  1.82550336,   2.17449664,   4.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  0.91275168,   1.08724832,   2.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  1.36912752,   1.63087248,   3.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  2.73825503,   3.26174497,   6.        ],
        [  8.67114094,  10.32885906,  19.        ],
        [  4.10738255,   4.89261745,   9.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  0.91275168,   1.08724832,   2.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  3.19463087,   3.80536913,   7.        ],
        [ 16.88590604,  20.11409396,  37.        ],
        [  1.82550336,   2.17449664,   4.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  2.73825503,   3.26174497,   6.        ],
        [  5.02013423,   5.97986577,  11.        ],
        [  1.36912752,   1.63087248,   3.        ],
        [  5.47651007,   6.52348993,  12.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [ 16.4295302 ,  19.5704698 ,  36.        ],
        [  3.65100671,   4.34899329,   8.        ],
        [  1.82550336,   2.17449664,   4.        ],
        [  2.73825503,   3.26174497,   6.        ],
        [  1.36912752,   1.63087248,   3.        ],
        [  5.93288591,   7.06711409,  13.        ],
        [ 13.69127517,  16.30872483,  30.        ],
        [  1.36912752,   1.63087248,   3.        ],
        [  0.91275168,   1.08724832,   2.        ],
        [  2.28187919,   2.71812081,   5.        ],
        [  0.91275168,   1.08724832,   2.        ],
        [  0.91275168,   1.08724832,   2.        ],
        [  7.30201342,   8.69798658,  16.        ],
        [  2.28187919,   2.71812081,   5.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  5.02013423,   5.97986577,  11.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  1.82550336,   2.17449664,   4.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  0.91275168,   1.08724832,   2.        ],
        [  1.36912752,   1.63087248,   3.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [136.        , 162.        , 298.        ]]))
```

Here p-value is 0.99 which is more than our 0.05 or acceptance level hence we fail to reject null hypothesis

So to conclude Anomalies in trestbps can detect heart attack

In [40]:
```
new_df.columns
```

Out[40]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target', 'Class'],
      dtype='object')
```

H0 = thalassemia is alone major cause of CVD

H1 = thalassemia is alone not a major cause of CVD

In [41]:
```
new_df_crosstab_thal = pd.crosstab(new_df['thal'],new_df['target'],margins=True)
new_df_crosstab_thal
```

Out[41]:

target	0	1	All
thal			
0	1	1	2
1	12	6	18
2	36	128	164
3	87	27	114
All	136	162	298

In [42]:
```
chi2_contingency(new_df_crosstab_thal)
```

Out[42]:
```
(83.55630018374487,
 9.380309421965574e-15,
 8,
 array([[  0.91275168,   1.08724832,   2.        ],
        [  8.2147651 ,   9.7852349 ,  18.        ],
        [ 74.84563758,  89.15436242, 164.        ],
        [ 52.02684564,  61.97315436, 114.        ],
        [136.        , 162.        , 298.        ]]))
```

Here we can see p-value is very less than 0.05 hence we reject null hypothesis and conclude that

thalassemia is alone not a major cause of CVD

Similarly checking for all the variables indivisually how much they can predict the CVD alone

In [43]:
```
new_df_crosstab_age = pd.crosstab(new_df['age'],new_df['target'],margins=True)
new_df_crosstab_age.head()
```

Out[43]:

target	0	1	All
age			
29	0	1	1
34	0	2	2
35	2	2	4
37	0	2	2
38	1	2	3

In [44]:
```
chi2_contingency(new_df_crosstab_age)
```

Out[44]:
```
(51.595175580873004,
 0.9965349093113395,
 82,
 array([[  0.45637584,   0.54362416,   1.        ],
        [  0.91275168,   1.08724832,   2.        ],
        [  1.82550336,   2.17449664,   4.        ],
        [  0.91275168,   1.08724832,   2.        ],
        [  1.36912752,   1.63087248,   3.        ],
        [  1.82550336,   2.17449664,   4.        ],
        [  1.36912752,   1.63087248,   3.        ],
        [  4.56375839,   5.43624161,  10.        ],
        [  3.65100671,   4.34899329,   8.        ],
        [  3.65100671,   4.34899329,   8.        ],
        [  5.02013423,   5.97986577,  11.        ],
        [  3.65100671,   4.34899329,   8.        ],
        [  3.19463087,   3.80536913,   7.        ],
        [  2.28187919,   2.71812081,   5.        ],
        [  3.19463087,   3.80536913,   7.        ],
        [  2.28187919,   2.71812081,   5.        ],
        [  3.19463087,   3.80536913,   7.        ],
        [  5.47651007,   6.52348993,  12.        ],
        [  5.93288591,   7.06711409,  13.        ],
        [  3.65100671,   4.34899329,   8.        ],
        [  7.30201342,   8.69798658,  16.        ],
        [  3.65100671,   4.34899329,   8.        ],
        [  4.56375839,   5.43624161,  10.        ],
        [  7.75838926,   9.24161074,  17.        ],
        [  8.67114094,  10.32885906,  19.        ],
        [  6.38926174,   7.61073826,  14.        ],
        [  5.02013423,   5.97986577,  11.        ],
        [  3.65100671,   4.34899329,   8.        ],
        [  4.56375839,   5.43624161,  10.        ],
        [  3.65100671,   4.34899329,   8.        ],
        [  4.56375839,   5.43624161,  10.        ],
        [  3.19463087,   3.80536913,   7.        ],
        [  3.19463087,   3.80536913,   7.        ],
        [  3.65100671,   4.34899329,   8.        ],
        [  1.82550336,   2.17449664,   4.        ],
        [  1.36912752,   1.63087248,   3.        ],
        [  1.82550336,   2.17449664,   4.        ],
        [  1.36912752,   1.63087248,   3.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [  0.45637584,   0.54362416,   1.        ],
        [136.        , 162.        , 298.        ]]))
```

In [45]:
```
new_df_crosstab_sex = pd.crosstab(new_df['sex'],new_df['target'])
new_df_crosstab_sex.head()
```

Out[45]:

target	0	1
sex		
0	22	69
1	114	93

In [46]:
```
chi2_contingency(new_df_crosstab_sex)
```

Out[46]:
```
(23.092443938363708,
 1.543962557935935e-06,
 1,
 array([[ 41.53020134,  49.46979866],
        [ 94.46979866, 112.53020134]]))
```

In [47]:
```
new_df_crosstab_cp = pd.crosstab(new_df['cp'],new_df['target'])
new_df_crosstab_cp.head()
```

Out[47]:

target	0	1
cp		
0	102	38
1	9	41
2	18	67
3	7	16

In [48]:
```
chi2_contingency(new_df_crosstab_cp)
```

Out[48]:
```
(79.84528830552777,
 3.312997960135616e-17,
 3,
 array([[63.89261745, 76.10738255],
        [22.81879195, 27.18120805],
        [38.79194631, 46.20805369],
        [10.4966443 , 12.5033557 ]]))
```

In [49]:
```
new_df_crosstab_chol = pd.crosstab(new_df['chol'],new_df['target'])
new_df_crosstab_chol.head()
```

Out[49]:

target	0	1
chol		
126	0	1
131	1	0
141	0	1
149	1	1
157	0	1

In [50]:
```
chi2_contingency(new_df_crosstab_chol)
```

Out[58]:
```
(168.1448408375696,
 0.10129406828057805,
 146,
 array([[0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [1.82550336, 2.17449664],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [2.73825503, 3.26174497],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [1.36912752, 1.63087248],
        [1.36912752, 1.63087248],
        [2.73825503, 3.26174497],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [1.82550336, 2.17449664],
        [2.28187919, 2.71812081],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [1.36912752, 1.63087248],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [1.82550336, 2.17449664],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [1.36912752, 1.63087248],
        [1.36912752, 1.63087248],
        [0.91275168, 1.08724832],
        [1.82550336, 2.17449664],
        [2.73825503, 3.26174497],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [1.82550336, 2.17449664],
        [1.82550336, 2.17449664],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [1.82550336, 2.17449664],
        [1.36912752, 1.63087248],
        [1.36912752, 1.63087248],
        [1.36912752, 1.63087248],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [2.28187919, 2.71812081],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [1.36912752, 1.63087248],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [2.28187919, 2.71812081],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [1.82550336, 2.17449664],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
```

```
       [0.45637584, 0.54362416],
       [0.45637584, 0.54362416],
       [0.91275168, 1.08724832],
       [1.36912752, 1.63087248],
       [0.45637584, 0.54362416],
       [0.45637584, 0.54362416],
       [0.91275168, 1.08724832],
       [0.91275168, 1.08724832],
       [0.45637584, 0.54362416],
       [0.45637584, 0.54362416],
       [0.45637584, 0.54362416],
       [0.91275168, 1.08724832],
       [0.45637584, 0.54362416],
       [0.45637584, 0.54362416],
       [0.91275168, 1.08724832],
       [0.91275168, 1.08724832],
       [0.45637584, 0.54362416],
       [0.45637584, 0.54362416],
       [0.45637584, 0.54362416],
       [0.45637584, 0.54362416],
       [0.45637584, 0.54362416],
       [0.45637584, 0.54362416]]))
```

In [51]:
```
new_df_crosstab_fbs = pd.crosstab(new_df['fbs'],new_df['target'])
chi2_contingency(new_df_crosstab_fbs)
```

Out[51]:
```
(0.21654924944208123,
 0.6416818903958912,
 1,
 array([[115.91946309, 138.08053691],
        [ 20.08053691,  23.91946309]]))
```

In [52]:
```
new_df_crosstab_restecg = pd.crosstab(new_df['restecg'],new_df['target'])
chi2_contingency(new_df_crosstab_restecg)
```

Out[52]:
```
(10.350736518833976,
 0.005654134342296864,
 2,
 array([[64.80536913, 77.19463087],
        [69.36912752, 82.63087248],
        [ 1.82550336,  2.17449664]]))
```

In [53]:
```
new_df_crosstab_thalach = pd.crosstab(new_df['thalach'],new_df['target'])
chi2_contingency(new_df_crosstab_thalach)
```

Out[53]:
```
(108.82953607556004,
 0.08613560428095296,
 90,
 array([[0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [1.36912752, 1.63087248],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [1.36912752, 1.63087248],
        [1.36912752, 1.63087248],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [1.82550336, 2.17449664],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [3.19463087, 3.80536913],
        [1.82550336, 2.17449664],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [1.82550336, 2.17449664],
        [1.82550336, 2.17449664],
        [3.19463087, 3.80536913],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [1.36912752, 1.63087248],
        [0.91275168, 1.08724832],
        [2.73825503, 3.26174497],
        [1.36912752, 1.63087248],
        [2.73825503, 3.26174497],
        [3.19463087, 3.80536913],
        [3.19463087, 3.80536913],
        [1.82550336, 2.17449664],
        [1.82550336, 2.17449664],
        [2.28187919, 2.71812081],
        [1.36912752, 1.63087248],
        [0.91275168, 1.08724832],
        [2.73825503, 3.26174497],
        [1.82550336, 2.17449664],
        [3.65100671, 4.34899329],
        [1.36912752, 1.63087248],
        [1.82550336, 2.17449664],
        [1.82550336, 2.17449664],
        [2.73825503, 3.26174497],
        [1.36912752, 1.63087248],
        [2.73825503, 3.26174497],
        [1.82550336, 2.17449664],
        [3.65100671, 4.34899329],
        [2.28187919, 2.71812081],
        [5.02013423, 5.97986577],
        [4.10738255, 4.89261745],
        [0.91275168, 1.08724832],
        [2.28187919, 2.71812081],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [2.28187919, 2.71812081],
        [2.73825503, 3.26174497],
        [2.28187919, 2.71812081],
        [1.82550336, 2.17449664],
        [3.19463087, 3.80536913],
        [3.65100671, 4.34899329],
        [2.28187919, 2.71812081],
        [1.36912752, 1.63087248],
        [0.45637584, 0.54362416],
        [2.28187919, 2.71812081],
        [2.28187919, 2.71812081],
        [0.91275168, 1.08724832],
        [0.91275168, 1.08724832],
        [2.28187919, 2.71812081],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.91275168, 1.08724832],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416],
        [0.45637584, 0.54362416]]))
```

In [54]:
```
new_df_crosstab_exang = pd.crosstab(new_df['exang'],new_df['target'])
chi2_contingency(new_df_crosstab_exang)
```

Out[54]:
```
(54.33094173317474,
 1.6941311974960348e-13,
 1,
 array([[ 91.27516779, 108.72483221],
        [ 44.72483221,  53.27516779]]))
```

In [55]:
```
new_df_crosstab_oldpeak = pd.crosstab(new_df['oldpeak'],new_df['target'])
chi2_contingency(new_df_crosstab_oldpeak)
```

Out[55]:
```
(88.3078180574368,
 1.0922859200974877e-05,
 39,
 array([[45.18120805, 53.81879195],
        [ 3.19463087,  3.80536913],
        [ 5.47651007,  6.52348993],
        [ 1.36912752,  1.63087248],
        [ 4.10738255,  4.89261745],
        [ 2.28187919,  2.71812081],
        [ 6.38926174,  7.61073826],
        [ 0.45637584,  0.54362416],
        [ 5.47651007,  6.52348993],
        [ 1.36912752,  1.63087248],
        [ 6.38926174,  7.61073826],
        [ 0.91275168,  1.08724832],
        [ 7.30201342,  8.69798658],
        [ 0.45637584,  0.54362416],
        [ 5.93288591,  7.06711409],
        [ 2.28187919,  2.71812081],
        [ 4.56375839,  5.43624161],
        [ 4.56375839,  5.43624161],
        [ 1.82550336,  2.17449664],
        [ 4.10738255,  4.89261745],
        [ 0.45637584,  0.54362416],
        [ 1.82550336,  2.17449664],
        [ 0.91275168,  1.08724832],
        [ 1.36912752,  1.63087248],
        [ 0.91275168,  1.08724832],
        [ 2.73825503,  3.26174497],
        [ 2.73825503,  3.26174497],
        [ 0.45637584,  0.54362416],
        [ 2.28187919,  2.71812081],
        [ 0.45637584,  0.54362416],
        [ 0.91275168,  1.08724832],
        [ 1.36912752,  1.63087248],
        [ 0.45637584,  0.54362416],
        [ 1.82550336,  2.17449664],
        [ 0.45637584,  0.54362416],
        [ 0.91275168,  1.08724832],
        [ 0.91275168,  1.08724832],
        [ 0.45637584,  0.54362416],
        [ 0.45637584,  0.54362416],
        [ 0.45637584,  0.54362416]]))
```

In [56]:
```
new_df_crosstab_slope = pd.crosstab(new_df['slope'],new_df['target'])
chi2_contingency(new_df_crosstab_slope)
```

Out[56]:
```
(47.242095913791594,
 5.514539038798198e-11,
 2,
 array([[ 9.58389262, 11.41610738],
        [62.06711409, 73.93288591],
        [64.34899329, 76.65100671]]))
```

In [57]:
```
new_df_crosstab_ca = pd.crosstab(new_df['ca'],new_df['target'])
chi2_contingency(new_df_crosstab_ca)
```

Out[57]:
```
(72.09317974110584,
 8.201875360183414e-15,
 4,
 array([[78.95302013, 94.04697987],
        [29.20805369, 34.79194631],
        [16.88590604, 20.11409396],
        [ 8.67114094, 10.32885906],
        [ 2.28187919,  2.71812081]]))
```

Now let us build a Logistic regression model using all the variables except class as it was only prepared for getting insights

In [58]:
```
features = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal']
x = new_df[features]
y = new_df['target']
```

In [59]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33)
```

In [60]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [61]:
```
logreg.fit(x_train,y_train)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[61]:
```
LogisticRegression()
```

In [62]:
```
y_pred = logreg.predict(x_test)
```

In [63]:
```
from sklearn.metrics import accuracy_score,recall_score,f1_score,precision_score
```

In [64]:
```
a = accuracy_score(y_test,y_pred)
a
```

Out[64]:
```
0.8686868686868687
```

In [65]:
```
p = precision_score(y_test,y_pred)
p
```

Out[65]:
```
0.8947368421052632
```

In [66]:
```
r = recall_score(y_test,y_pred)
r
```

Out[66]:
```
0.8793103448275862
```

In [67]:
```
f = f1_score(y_test,y_pred)
f
```

Out[67]:
```
0.8869565217391304
```

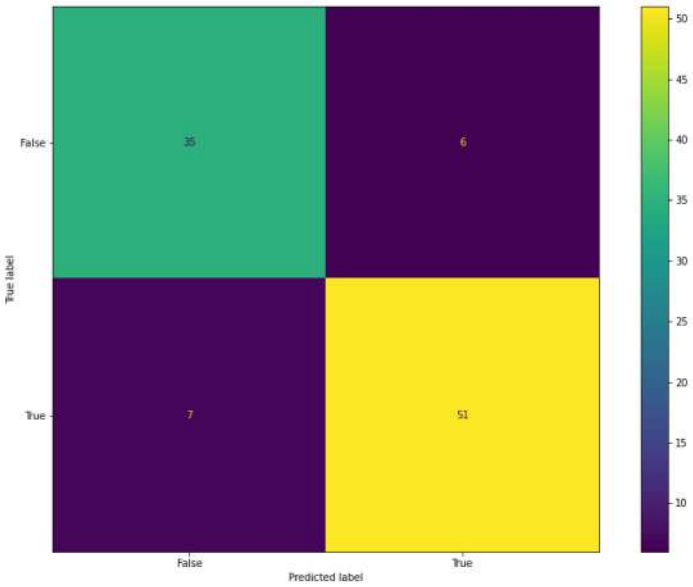
In [68]:
```
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test,y_pred)
cm1
```

Out[68]:
```
array([[35,  6],
       [ 7, 51]], dtype=int64)
```

In [69]:
```
from sklearn.metrics import ConfusionMatrixDisplay
cd = ConfusionMatrixDisplay
```

In [70]:
```
conf_matrix1 = cd(confusion_matrix=cm1,display_labels=[False,True])
conf_matrix1.plot()
```

Out[70]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x215773d0e80>
```



We have Built a very accurate model with accuracy score of 86.87%

Now we will build a random forest model to see if we can achieve more accuracy than LOGISTIC REGRESSION Model

In [71]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=800)
```

In [72]:
```
rf.fit(x_train,y_train)
```

Out[72]:
```
RandomForestClassifier(n_estimators=800)
```

In [73]:
```
y_pred1 = rf.predict(x_test)
```

In [74]:
```
a2 = accuracy_score(y_test,y_pred1)
a2
```

Out[74]:
```
0.8484848484848485
```

In [75]:
```
p2 = precision_score(y_test,y_pred1)
p2
```

Out[75]:
```
0.8771929824561403
```

In [76]:
```
r2 = recall_score(y_test,y_pred1)
r2
```

Out[76]:
```
0.8620689655172413
```

In [77]:
```
f2 = f1_score(y_test,y_pred1)
f2
```

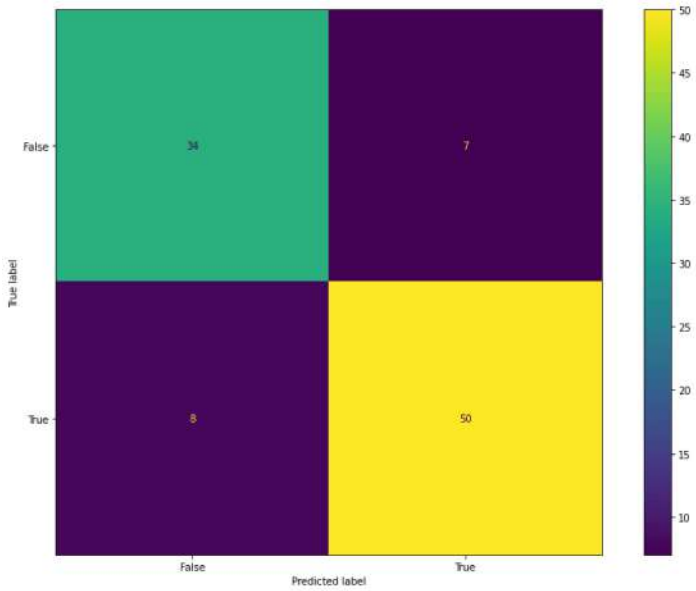
Out[77]:
```
0.8695652173913043
```

In [78]:
```
cm2 = confusion_matrix(y_test,y_pred1)
cm2
```

Out[78]:
```
array([[34,  7],
       [ 8, 50]], dtype=int64)
```

In [79]:
```
conf_matrix2 = cd(confusion_matrix=cm2,display_labels=[False,True])
conf_matrix2.plot()
```

Out[79]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x215777d9dc0>
```



We are getting somewhat similar accuracy for both the models that is around 84% to 86%

We have checked whether indivisual parameters affect the CVD

Based on p-values obtained from the previous tests we will select only important features and check if we can improve the accuracy of our model

In [80]:
```
new_df.columns
```

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

In [81]:
```
imp_features = ['age','trestbps','chol','fbs','thalach','restecg','oldpeak']
x1 = new_df[imp_features]
y1 = new_df['target']
```

In [82]:
```
x1_train,x1_test,y1_train,y1_test = train_test_split(x1,y1,test_size=0.33)
```

In [83]:
```
rf.fit(x1_train,y1_train)
```

Out[83]:
```
RandomForestClassifier(n_estimators=800)
```

In [84]:
```
y_pred2 = rf.predict(x1_test)
```

In [85]:
```
a3 = accuracy_score(y1_test,y_pred2)
a3
```

Out[85]:
```
0.6868686868686869
```

In [86]:
```
p3 = precision_score(y1_test,y_pred2)
p3
```

Out[86]:
```
0.6666666666666666
```

In [87]:
```
r3 = recall_score(y1_test,y_pred2)
r3
```

Out[87]:
```
0.76
```

In [88]:
```
f3 = f1_score(y1_test,y_pred2)
f3
```

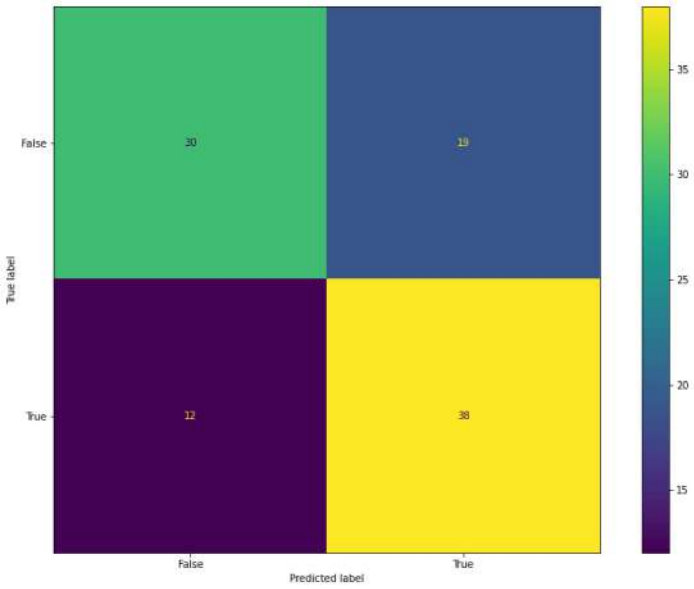
Out[88]:
```
0.7102803738317756
```

In [89]:
```
cm3 = confusion_matrix(y1_test,y_pred2)
cm3
```

Out[89]:
```
array([[30, 19],
       [12, 38]], dtype=int64)
```

In [90]:
```
conf_matrix3 = cd(confusion_matrix=cm3,display_labels=[False,True])
conf_matrix3.plot()
```

Out[90]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2157780dee0>
```



We can clearly see the drop in accuracy which shows that more features required for predicting CVD presence for a patient hence we will continue with previous models

Thank You