Checking for outliers using IQR

[14]:

```
# outlier detection using IQR
for column in df_heart[choosen_features_nums]:
    for target in df_heart['target'].unique():
        q25 = df_heart[column][df_heart['target'] == target].quantile(0.25)
        q75 = df_heart[column][df_heart['target'] == target].quantile(0.75)
        iqr = q75 - q25
        print(target, '-', column.upper())
        print('Percentiles: 25th = %.3f, 75th = %.3f, IQR = %.3f' % (q25, q75, iqr))

        # Calculate the outlier cutoff
        cut_off = iqr * 1.5
        lower, upper = q25 - cut_off, q75 + cut_off

        # Identify outliers
        df_heart2 = pd.DataFrame(df_heart[df_heart['target'] == target][column])

        count = len(df_heart2[df_heart2[column] < lower].index)
        count += len(df_heart2[df_heart2[column] > upper].index)
        print('Identified outliers: ', count)

        # replacing outliers with NaN (Will be later replaced with feature mean)
        for index in df_heart2[df_heart2[column] < lower].index:
            df_heart.loc[index, column] = np.nan
        for index in df_heart2[df_heart2[column] > upper].index:
            df_heart.loc[index, column] = np.nan
```

```
1 - OLDPEAK
Percentiles: 25th = -0.023, 75th = 0.906, IQR = 0.929
Identified outliers:  5
0 - OLDPEAK
Percentiles: 25th = 0.623, 75th = 2.555, IQR = 1.932
Identified outliers:  2
1 - THALACH
Percentiles: 25th = 148.052, 75th = 172.048, IQR = 23.996
Identified outliers:  1
0 - THALACH
Percentiles: 25th = 124.972, 75th = 156.158, IQR = 31.186
Identified outliers:  0
```

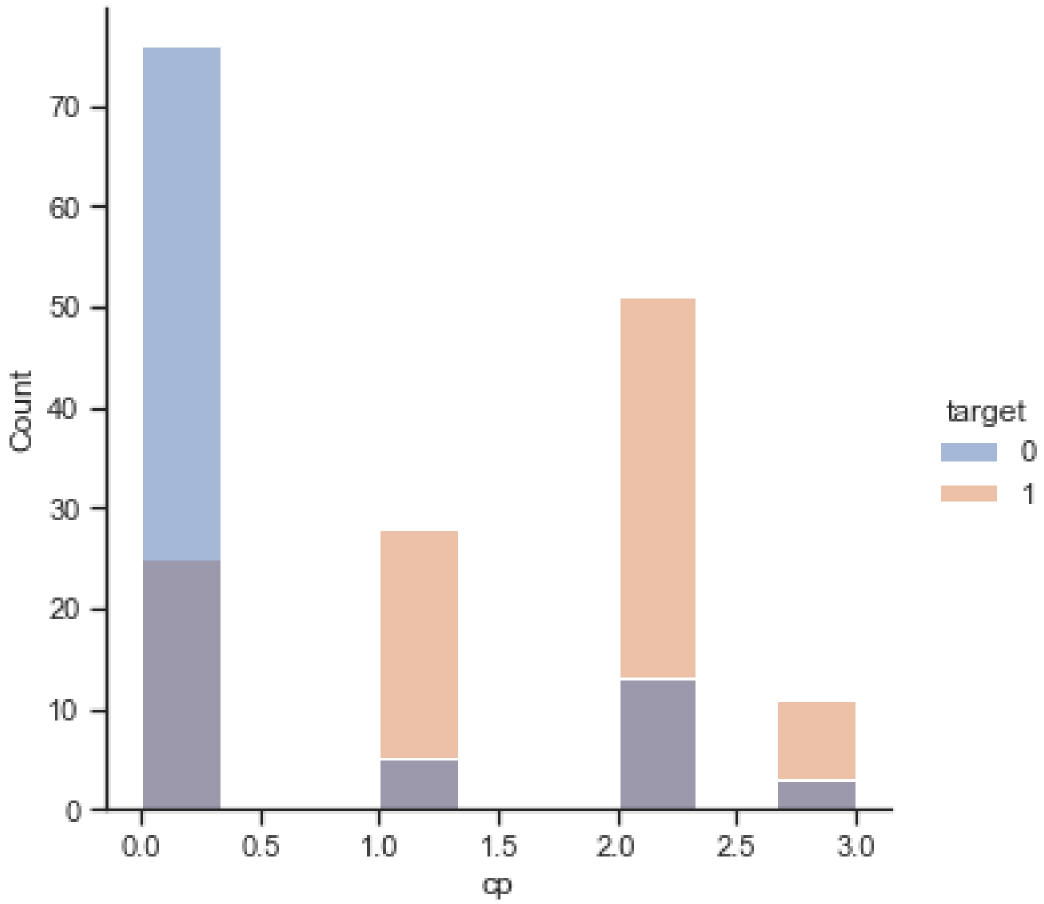
We observe that the number of outliers found corresponds to the box-plot. These outliers can be handled by replacing with feature mean.

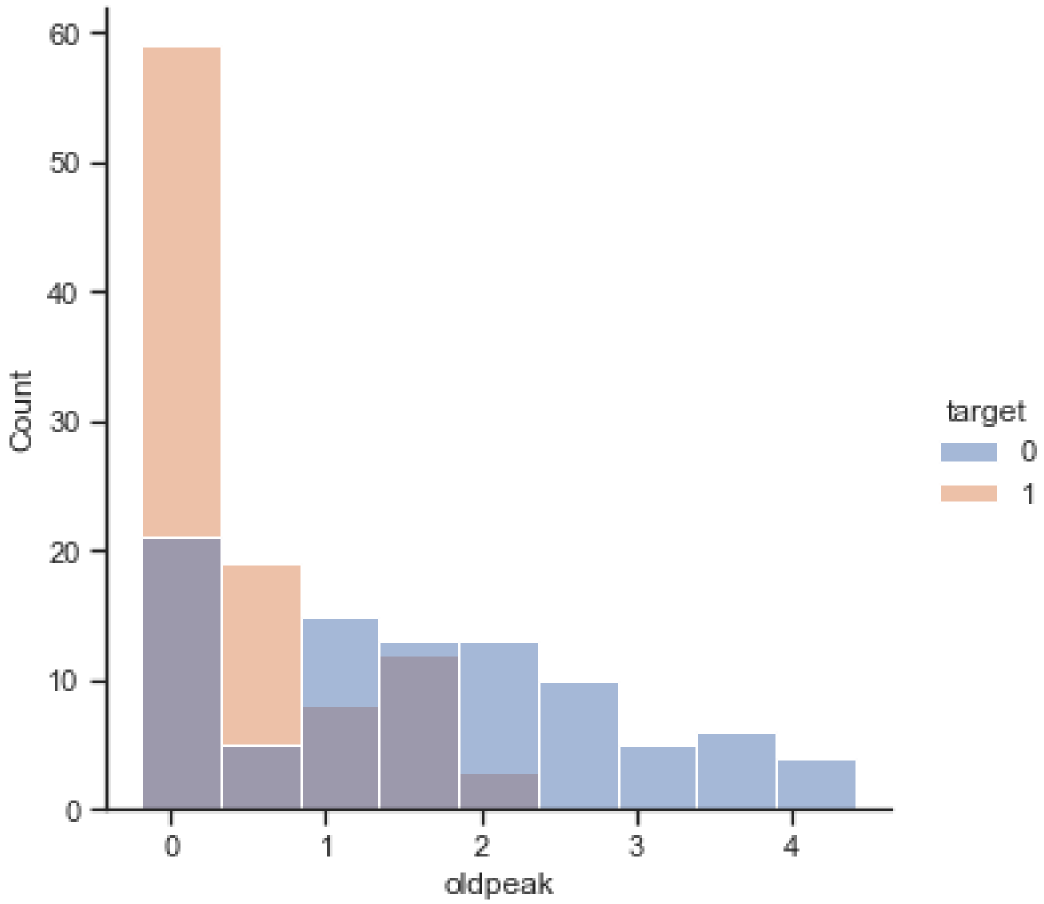
[CM4]

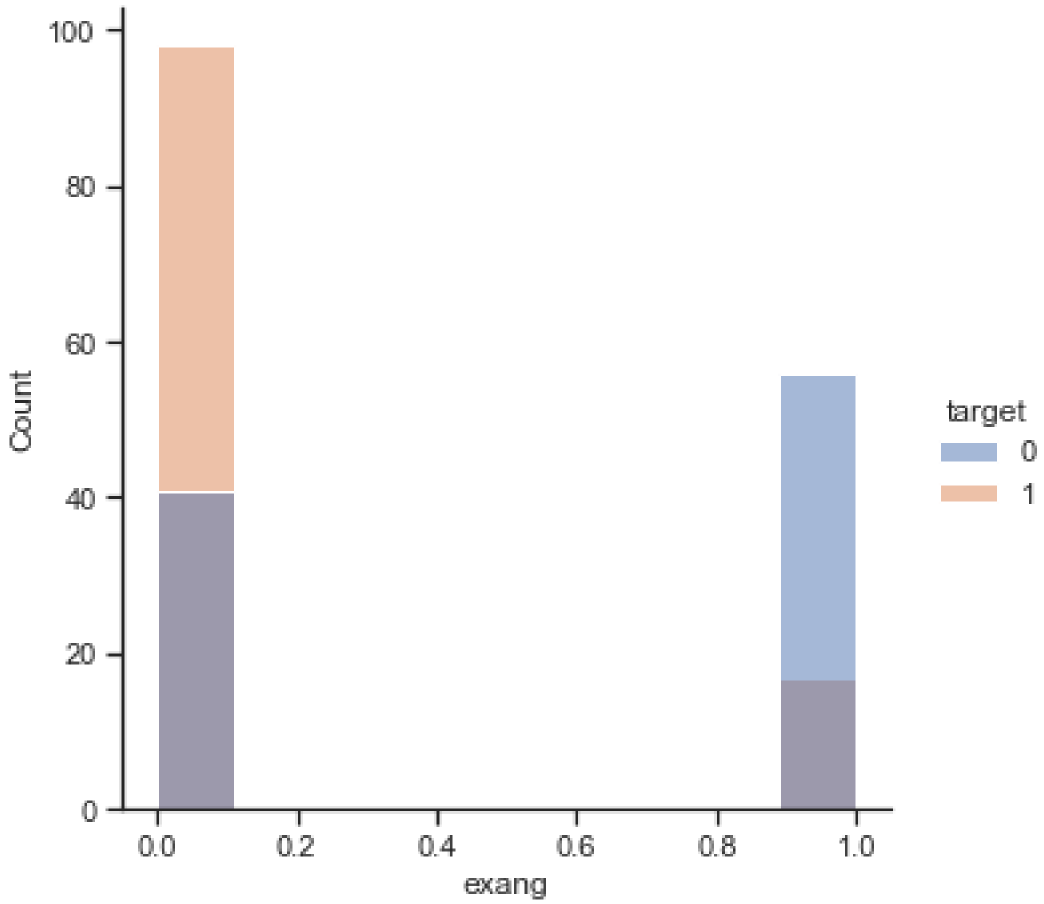
Histogram plot of the features

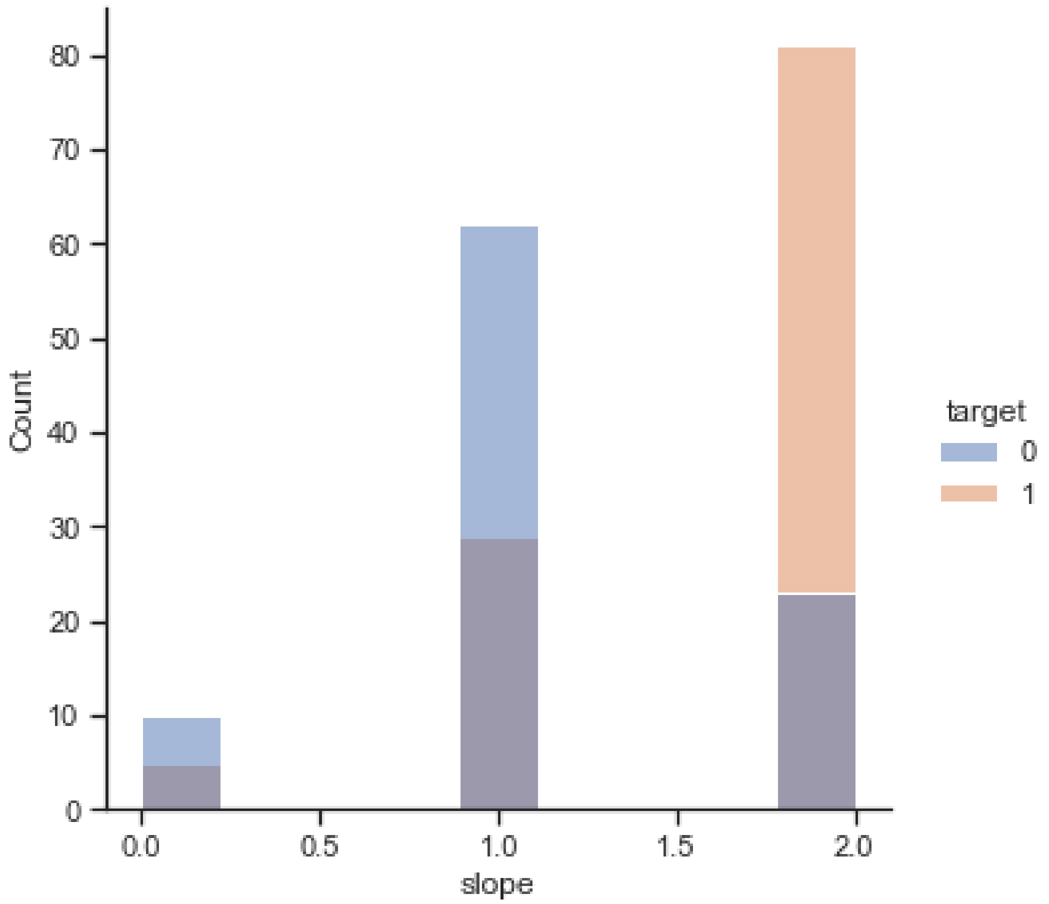
[15]:

```
# plot histogram
for column in df_heart[choosen_features]:
    sns.displot(df_heart, x=column, hue="target")
```

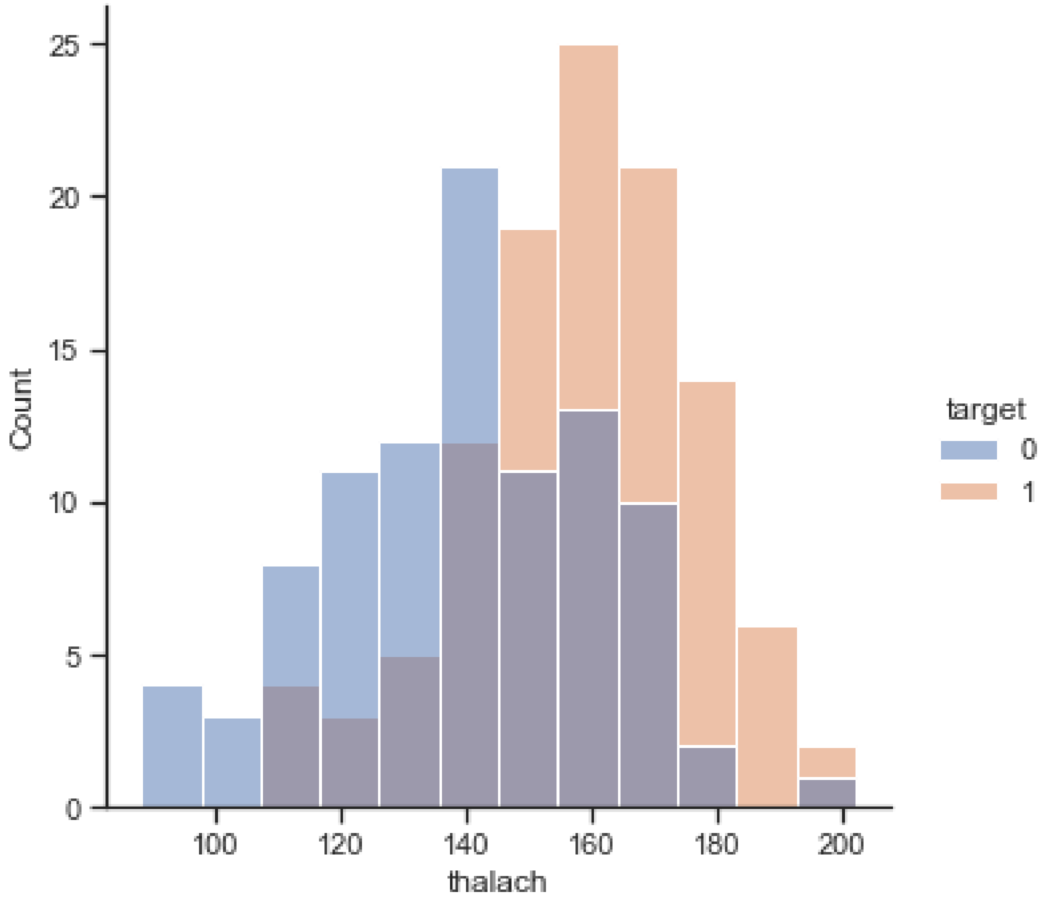

Count
0
10
20
30
40
50
60
70
0.0
0.5
1.0
1.5
2.0
2.5
3.0
cp
target
0
1


Count
60
50
40
30
20
10
0
0
1
2
3
4
oldpeak
target
0
1


0
20
40
60
80
100
Count
0.0
0.2
0.4
0.6
0.8
1.0
exang
target
0
1



80
70
60
50
40
30
20
10
0
Count
0.0
0.5
1.0
1.5
2.0
slope
target
0
1



From the above histograms, we can see the number of present (1) and absent (0) heart disease cases in each features.

[CM5]

Data Cleaning

Checking for null / NaN values (missing data)

[16]:
```
# checking for any null / NaN values
df_heart.isnull().values.any()
```

[16]:
```
True
```

[17]:
```
# checking for any null / NaN values
df_heart.isna().sum()
```

[17]:
```
age          0
sex          0
cp           0
trestbps     7
chol        10
fbs          0
```