In [357]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("seaborn-bright")
```

In [358]:
```
df = pd.read_excel('1645792390_cep1_dataset.xlsx')
```

In [359]:
```
df.head(20)
```

Out[359]:

	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
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
12	49	1	1	130	266	0	1	171	0	0.6	2	0	2	1
13	64	1	3	110	211	0	0	144	1	1.8	1	0	2	1
14	58	0	3	150	283	1	0	162	0	1.0	2	0	2	1
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2	1
16	58	0	2	120	340	0	1	172	0	0.0	2	0	2	1
17	66	0	3	150	226	0	1	114	0	2.6	0	0	2	1
18	43	1	0	150	247	0	1	171	0	1.5	2	0	2	1
19	69	0	3	140	239	0	1	151	0	1.8	2	2	2	1

In [360]:
```
df.shape
```

Out[360]: (303, 14)

In [361]:
```
df.info()
#No missing values
```

```
<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 [362]:
```
df.describe().T
```

Out[362]:

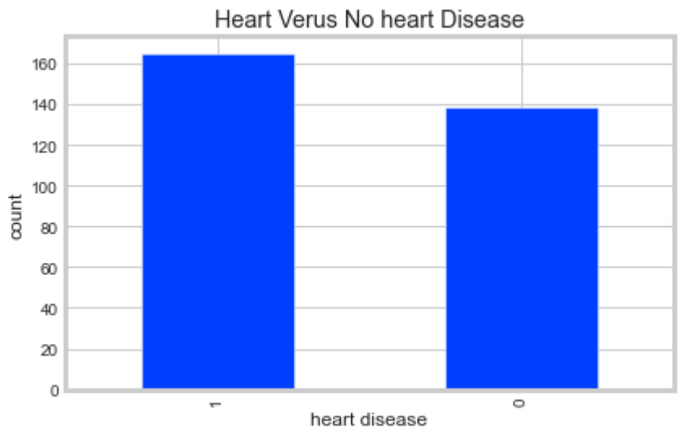
	count	mean	std	min	25%	50%	75%	max
age	303.0	54.366337	9.082101	29.0	47.5	55.0	61.0	77.0
sex	303.0	0.683168	0.466011	0.0	0.0	1.0	1.0	1.0
cp	303.0	0.966997	1.032052	0.0	0.0	1.0	2.0	3.0
trestbps	303.0	131.623762	17.538143	94.0	120.0	130.0	140.0	200.0
chol	303.0	246.264026	51.830751	126.0	211.0	240.0	274.5	564.0
fbs	303.0	0.148515	0.356198	0.0	0.0	0.0	0.0	1.0
restecg	303.0	0.528053	0.525860	0.0	0.0	1.0	1.0	2.0
thalach	303.0	149.646865	22.905161	71.0	133.5	153.0	166.0	202.0
exang	303.0	0.326733	0.469794	0.0	0.0	0.0	1.0	1.0
oldpeak	303.0	1.039604	1.161075	0.0	0.0	0.8	1.6	6.2
slope	303.0	1.399340	0.616226	0.0	1.0	1.0	2.0	2.0
ca	303.0	0.729373	1.022606	0.0	0.0	0.0	1.0	4.0
thal	303.0	2.313531	0.612277	0.0	2.0	2.0	3.0	3.0
target	303.0	0.544554	0.498835	0.0	0.0	1.0	1.0	1.0

In [363]:
```
df['target'].value_counts()
```

Out[363]:
```
1    165
0    138
Name: target, dtype: int64
```

In [364]:
```
df['target'].value_counts().plot.bar(title = "Heart Verus No heart Disease",xlabel='heart disease',ylabel= 'count')
```

Out[364]:
```
<AxesSubplot:title={'center':'Heart Verus No heart Disease'}, xlabel='heart disease', ylabel='count'>
```



In [365]:
```
#checking for the missing values

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

Out[365]:
```
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 [366]:
```
df_cat = []
df_num=[]
for column in df.columns:
    if len(df[column].unique())<=10:
        df_cat.append(column)
    else:
        df_num.append(column)
```

In [367]:
```
df_cat
```

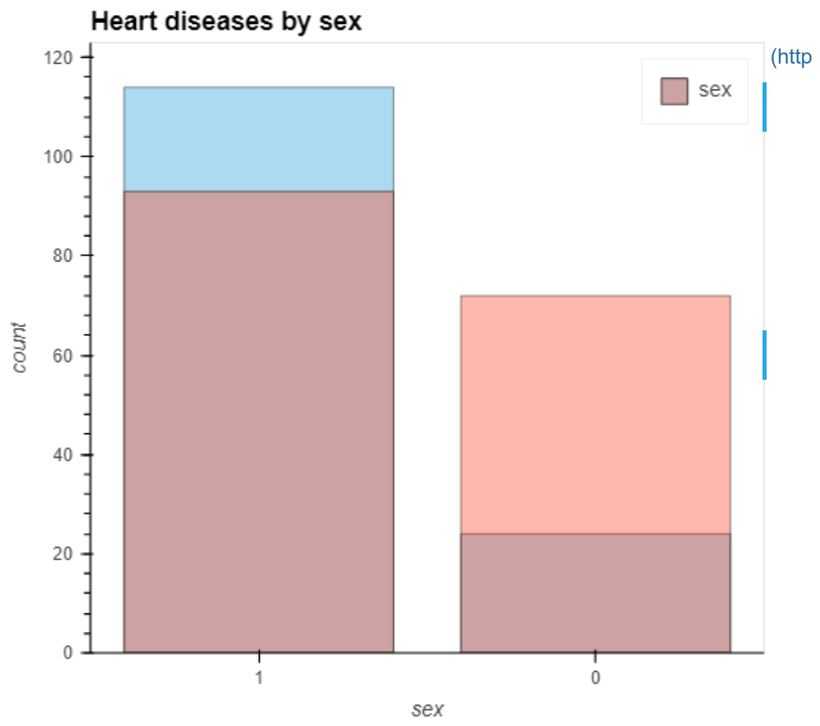
Out[367]:
```
['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
```

In [368]:
```
df_num
```

Out[368]:
```
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

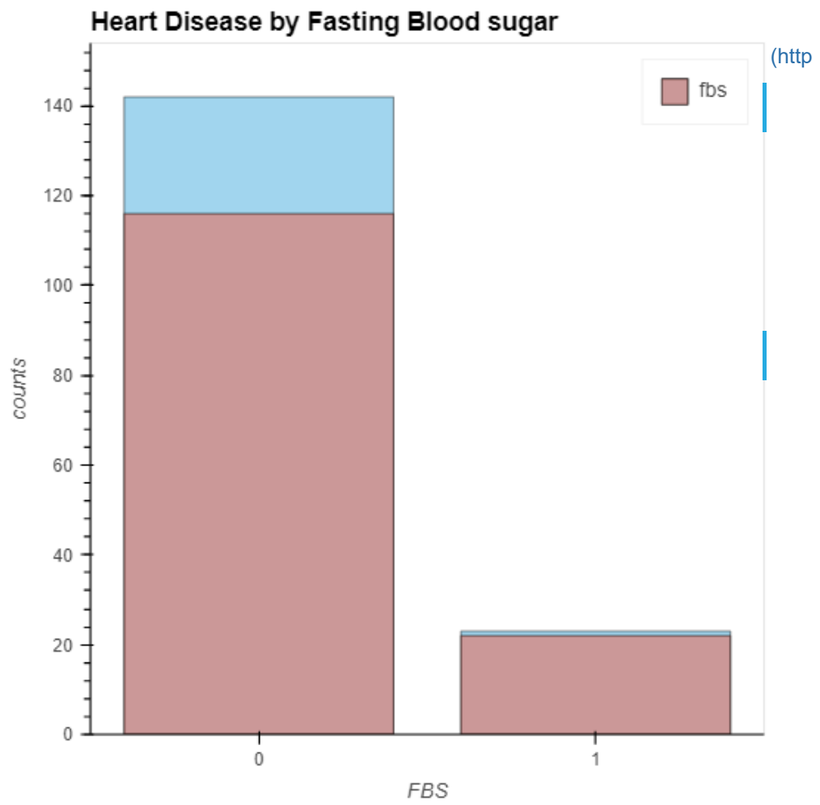
In [369]:
```
import hvplot.pandas
df_sex_0 = df.loc[df['target']==0,'sex'].value_counts().hvplot.bar(alpha=0.4)
df_sex_1 = df.loc[df['target']==1,'sex'].value_counts().hvplot.bar(alpha=0.4)

(df_sex_0*df_sex_1).opts(title='Heart diseases by sex',xlabel='sex',ylabel='count',width=500,height=450,legend_cols=2,
                         legend_position='top_right')
```

Out[369]:



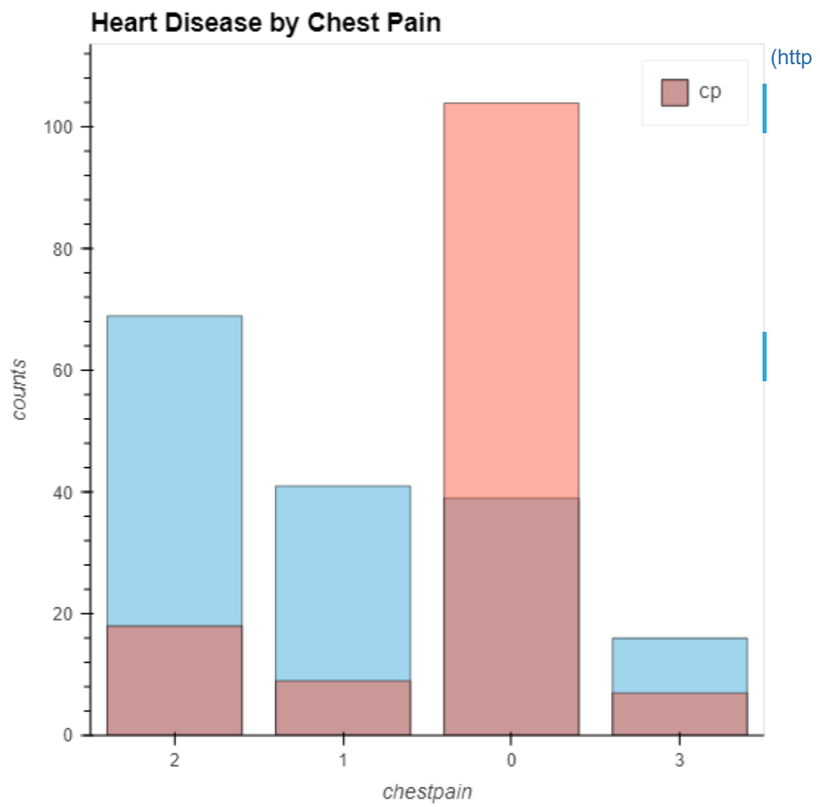
In [485]:
```
df_fbs_0 = df.loc[df['target']==0,'fbs'].value_counts().hvplot.bar(alpha=0.45)
df_fbs_1 = df.loc[df['target']==1,'fbs'].value_counts().hvplot.bar(alpha=0.45)

(df_fbs_1*df_fbs_0).opts(title = 'Heart Disease by Fasting Blood sugar',xlabel='FBS',ylabel='counts',width = 500,height=500,
                         legend_cols=1,legend_position='top_right')
```

Out[485]:



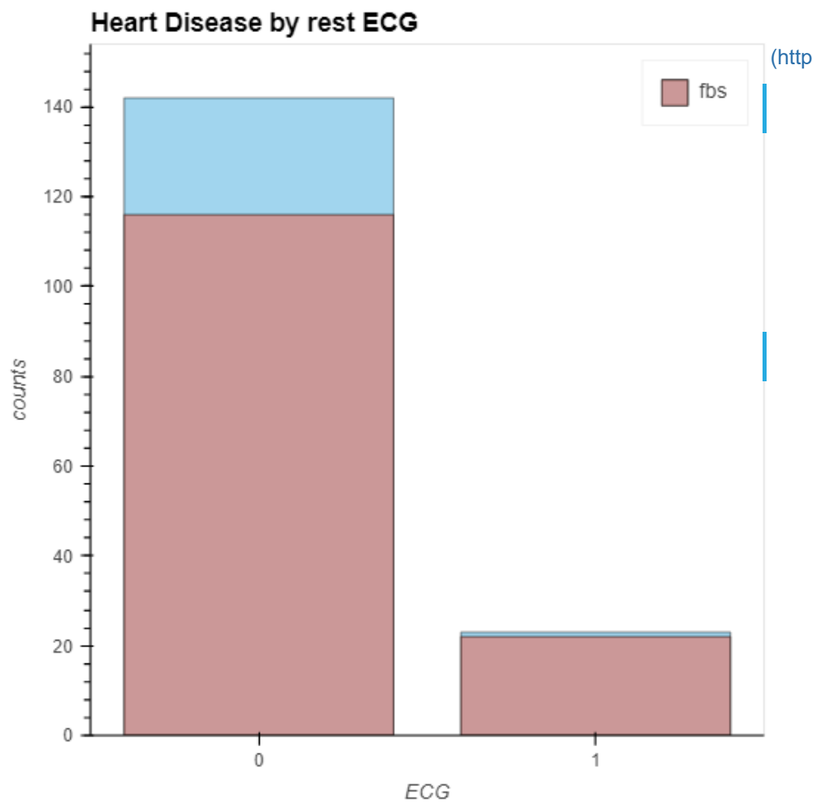
In [486]:
```
df_cp_0 = df.loc[df['target']==0,'cp'].value_counts().hvplot.bar(alpha=0.45)
df_cp_1 = df.loc[df['target']==1,'cp'].value_counts().hvplot.bar(alpha=0.45)

(df_cp_1*df_cp_0).opts(title = 'Heart Disease by Chest Pain',xlabel='chestpain',ylabel='counts',width = 500,height=500,
                       legend_cols=1,legend_position='top_right')
```

Out[486]:



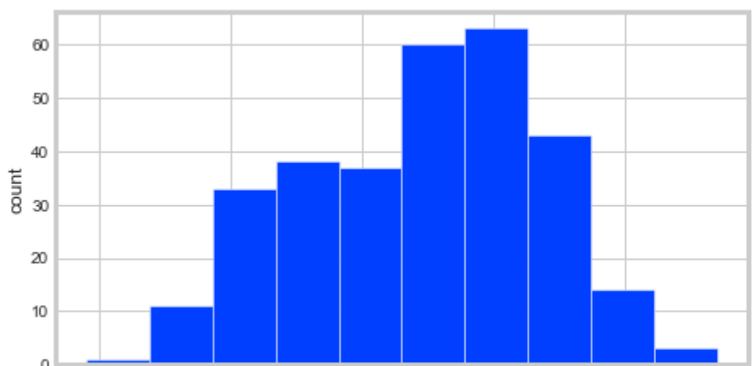
In [487]:
```
df_rest_0 = df.loc[df['target']==0,'fbs'].value_counts().hvplot.bar(alpha=0.45)
df_rest_1 = df.loc[df['target']==1,'fbs'].value_counts().hvplot.bar(alpha=0.45)

(df_rest_1*df_fbs_0).opts(title = 'Heart Disease by rest ECG',xlabel='ECG',ylabel='counts',width = 500,height=500,
                          legend_cols=1,legend_position='top_right')
```

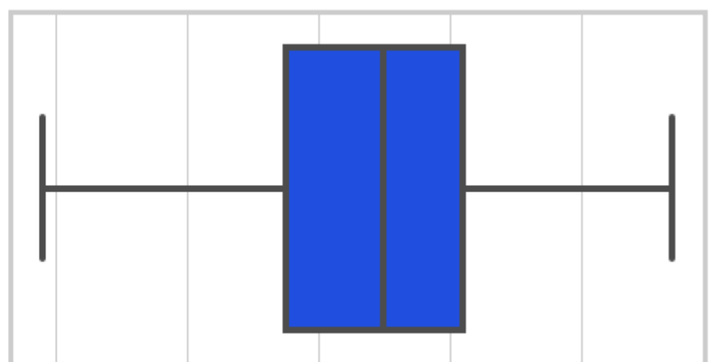
Out[487]:



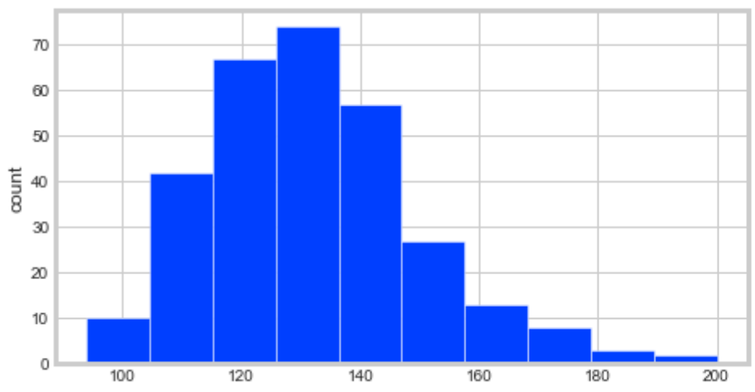
In [474]:
```
#univariate analysis

for col in df[df_num].columns:
    print(col)
    print("skew: ",round(df[col].skew(),2))
    plt.figure(figsize=(15,4))
    plt.subplot(1,2,1)
    df[col].hist()
    plt.ylabel('count')
    plt.subplot(1,2,2)
    sns.boxplot(x=df[col])
    plt.show()
```

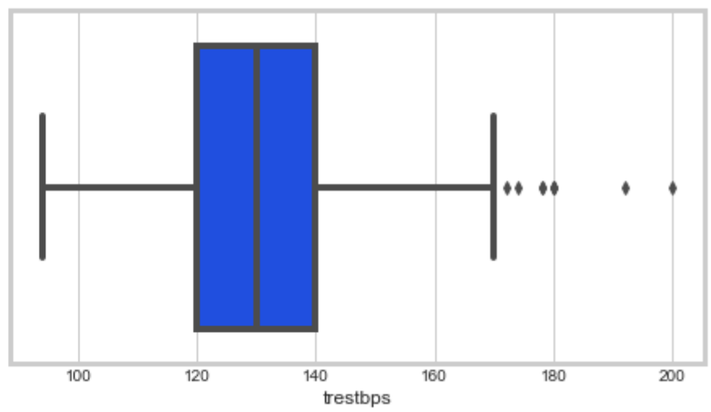
```
age
skew:  -0.2
```



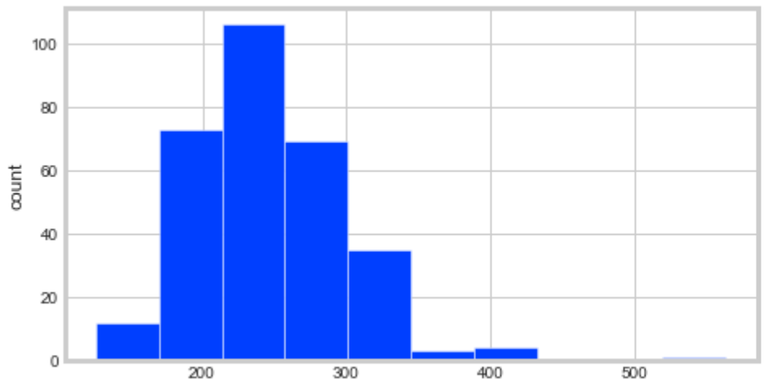


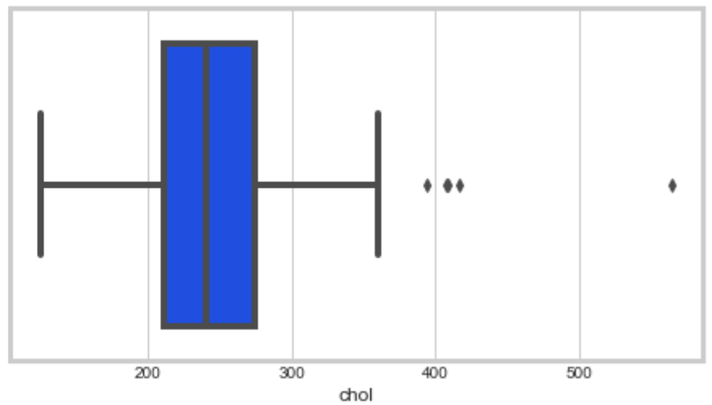

```
trestbps
skew:  0.71
```



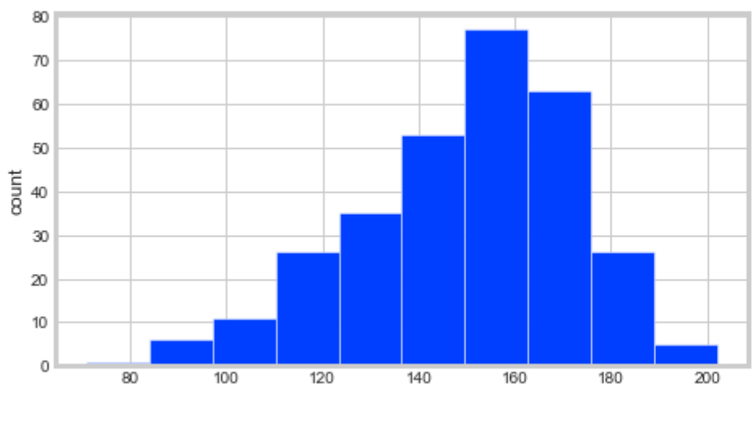


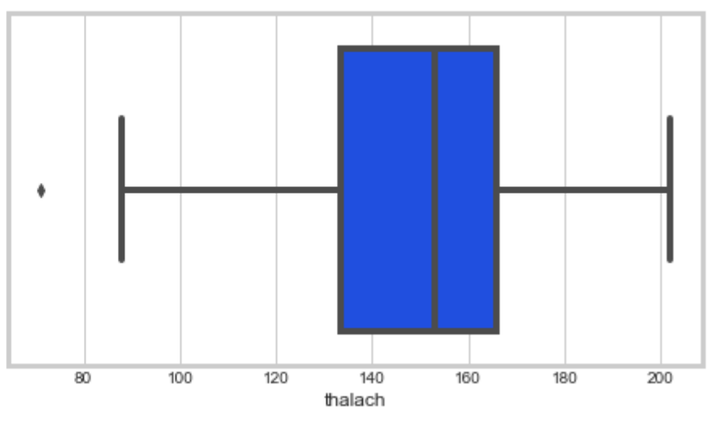

```
chol
skew:  1.14
```



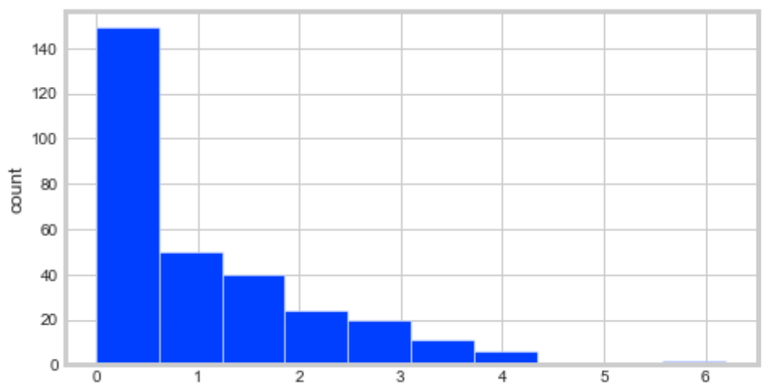


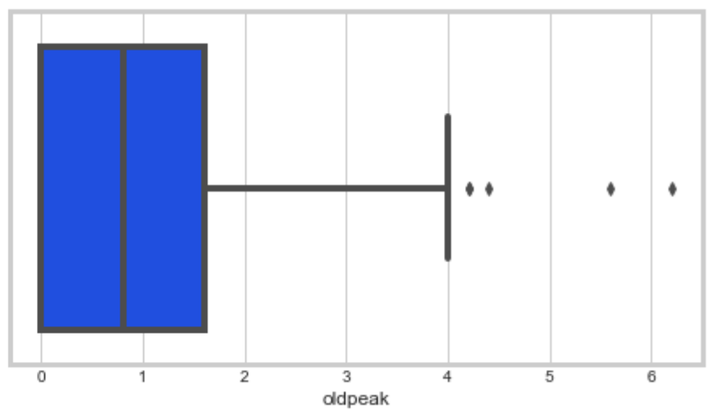

```
thalach
skew:  -0.54
```






```
oldpeak
skew:  1.27
```





In [476]:
```
df_catg = df[df_cat]
```

In [477]:
```
df_catg
```

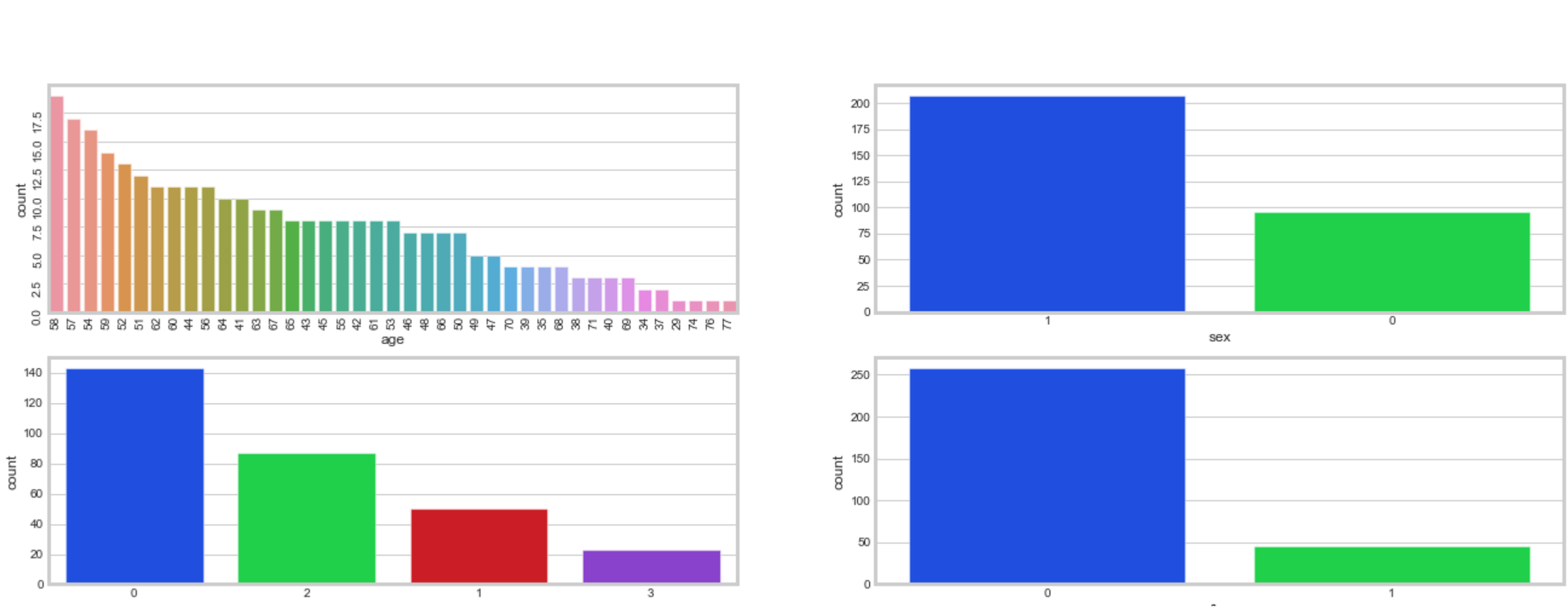
Out[477]:

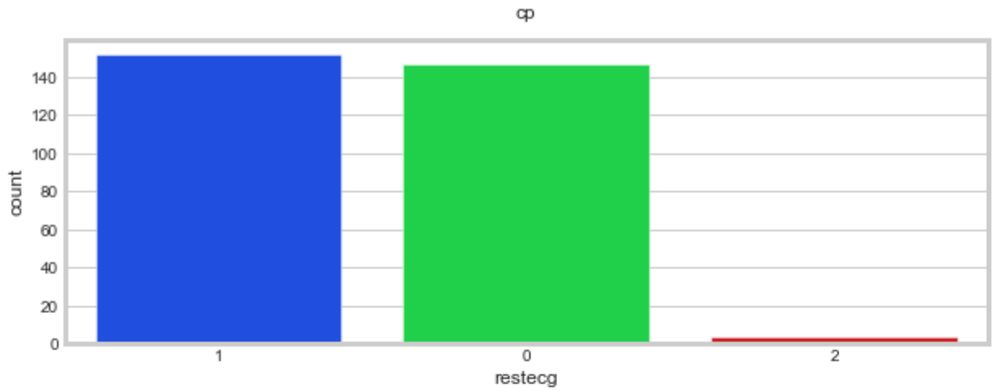
	sex	cp	fbs	restecg	exang	slope	ca	thal
0	1	3	1	0	0	0	0	1
1	1	2	0	1	0	0	0	2
2	0	1	0	0	0	2	0	2
3	1	1	0	1	0	2	0	2
4	0	0	0	1	1	2	0	2
...	...	...	...	...	...	...	...	...
298	0	0	0	1	1	1	0	3
299	1	3	0	1	0	1	0	3
300	1	0	1	1	0	1	2	3
301	1	0	0	1	1	1	1	3
302	0	1	0	0	0	1	1	2

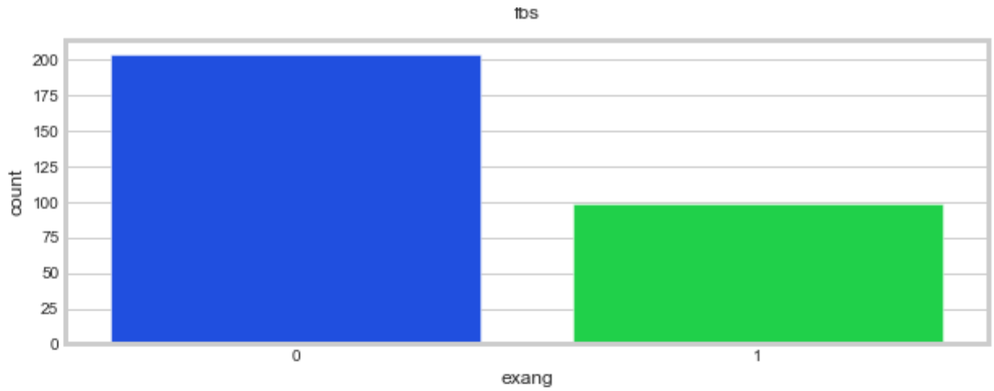
303 rows × 8 columns

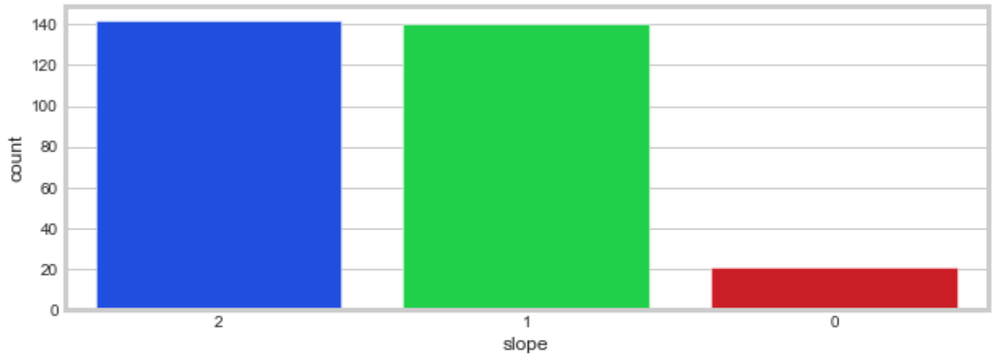
In [483]:
```
fig,axes=plt.subplots(5,2,figsize=(20,20))

fig.suptitle("data analysis on the categorical variables")

sns.countplot(x='age',order=df['age'].value_counts().index,data=df,ax= axes[0,0])
axes[0][0].tick_params(labelrotation=90)


sns.countplot(x='sex',order=df_catg['sex'].value_counts().index,data=df_catg,ax=axes[0,1])

sns.countplot(x='cp',data=df_catg,order=df_catg['cp'].value_counts().index,ax=axes[1,0])

sns.countplot(x='fbs',data=df_catg,order=df_catg['fbs'].value_counts().index,ax=axes[1,1])

sns.countplot(x='restecg',data=df_catg,order=df_catg['restecg'].value_counts().index,ax=axes[2,0])

sns.countplot(x='exang',data=df_catg,order=df_catg['exang'].value_counts().index,ax=axes[2,1])

sns.countplot(x='slope',data=df_catg,order=df_catg['slope'].value_counts().index,ax=axes[3,0])

sns.countplot(x='ca',data=df_catg,order=df_catg['ca'].value_counts().index,ax=axes[3,1])

sns.countplot(x='thal',data=df_catg,order=df_catg['thal'].value_counts().index,ax=axes[4,0])
```

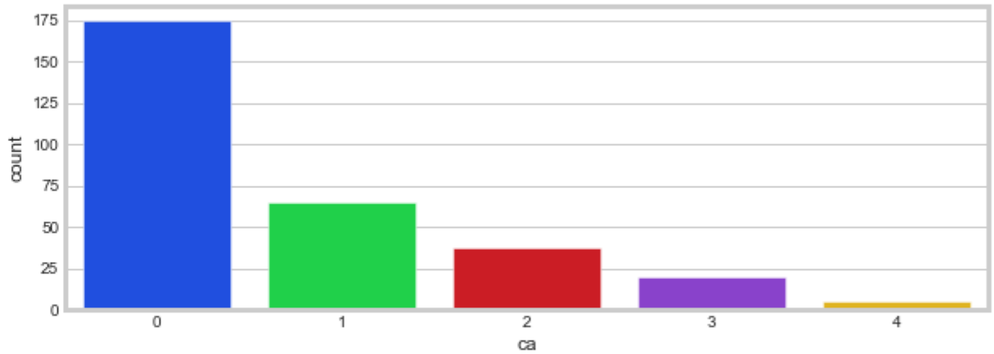
Out[483]: <AxesSubplot:xlabel='thal', ylabel='count'>

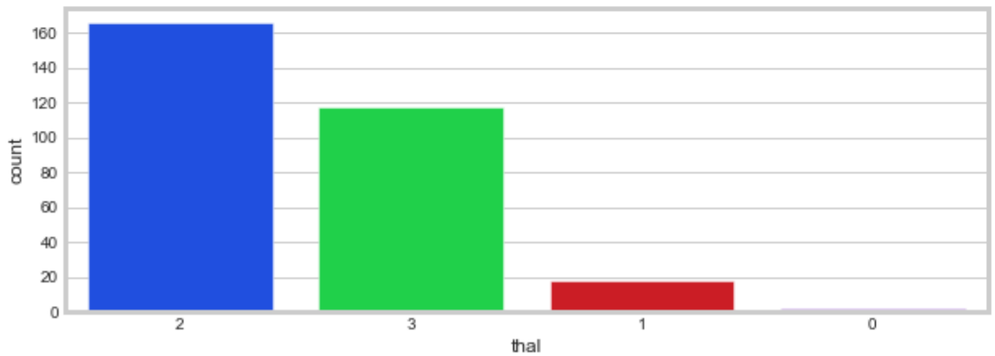


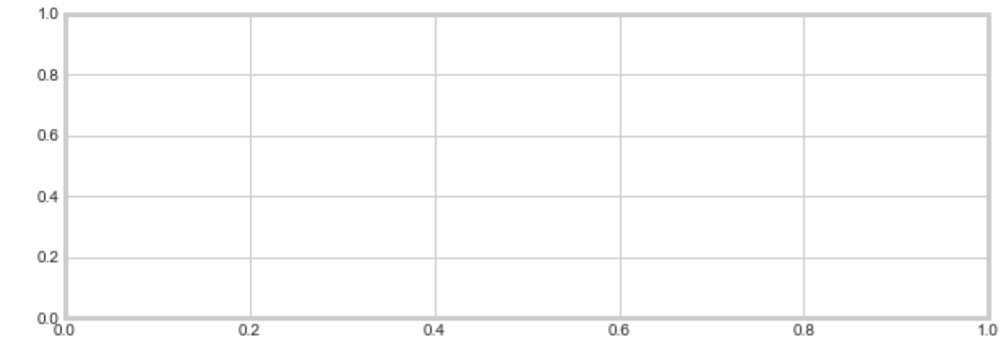

cp
count
restecg


fbs
count
exang

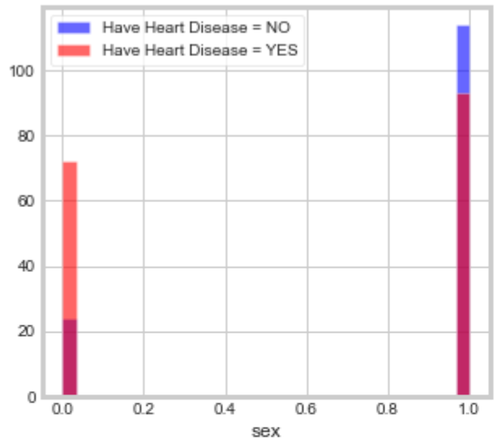

count
slope

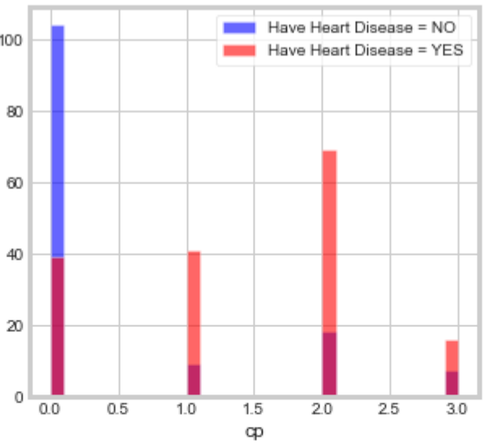

count
ca

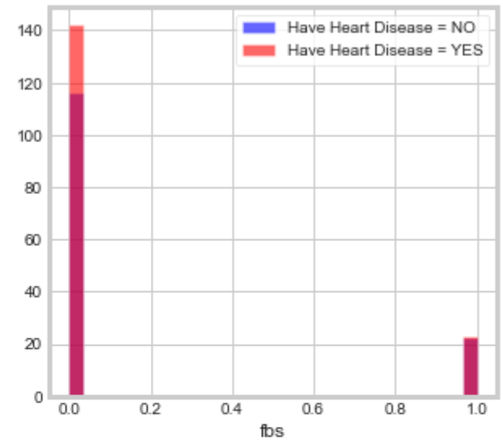

count
thal

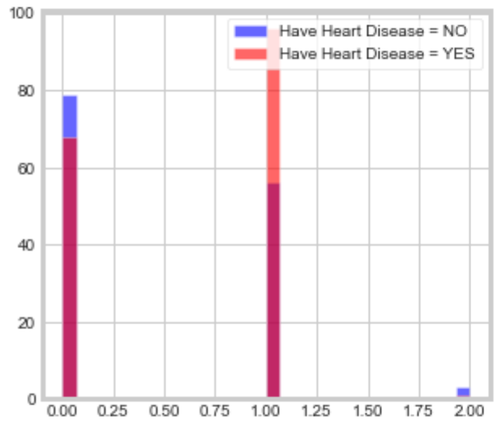


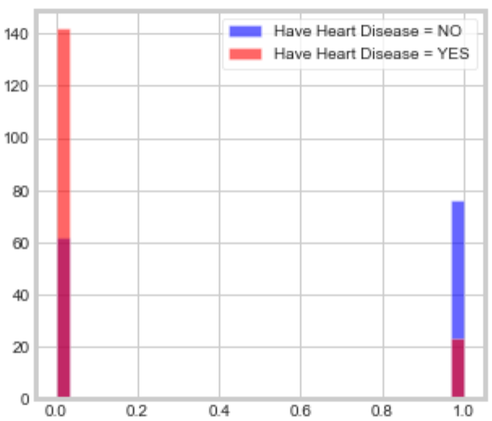
In [371]:
```
#Bi variate Analysis

plt.figure(figsize=(15,15))

for i,column in enumerate(df_cat,1):
    plt.subplot(3,3,i)
    df[df['target']==0][column].hist(bins=30,color='blue',label='Have Heart Disease = NO',alpha=0.6)
    df[df['target']==1][column].hist(bins=30,color='red',label='Have Heart Disease = YES',alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```

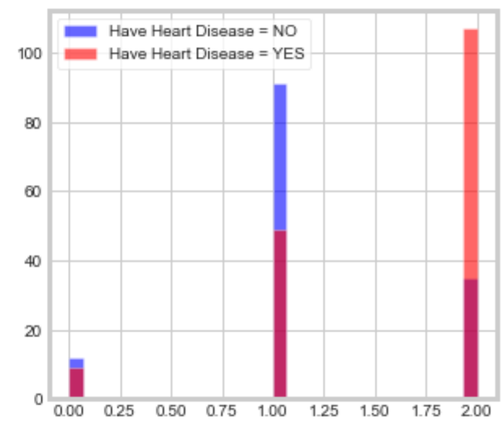


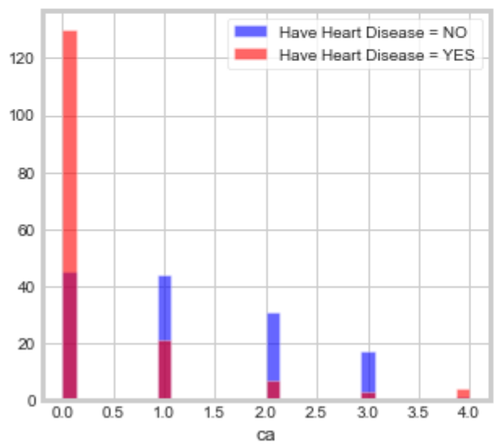


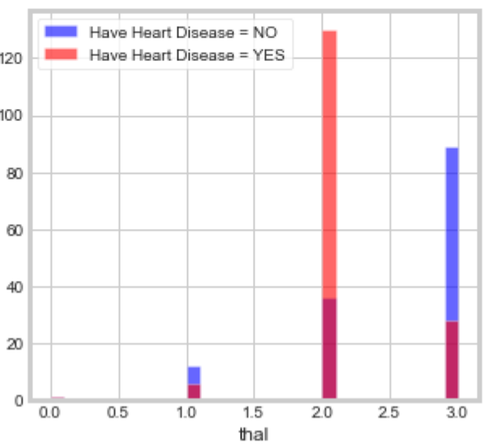


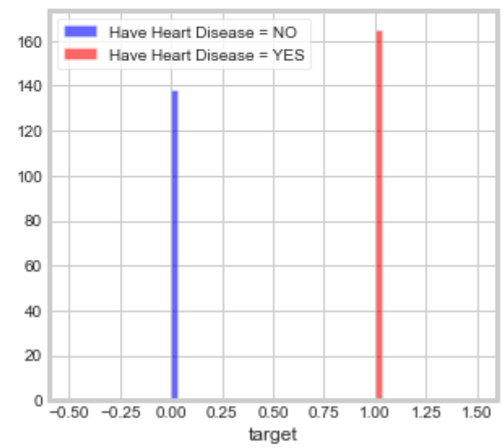












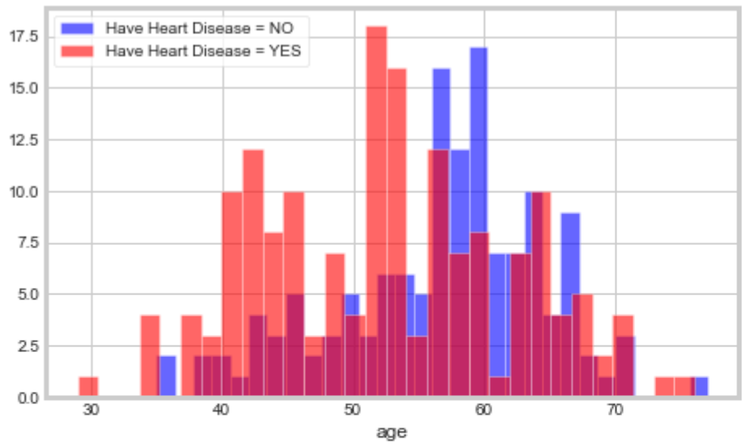
Major Insights Drawn from Above:

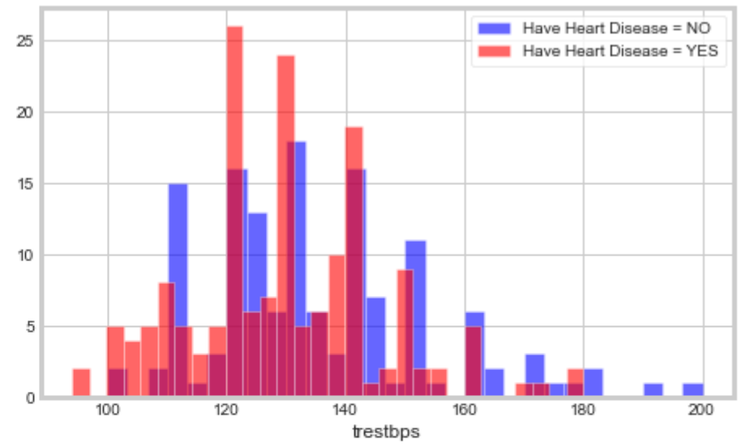
1) people with chestpain of 1,2,3 are more often have been diagonised with the heart disease

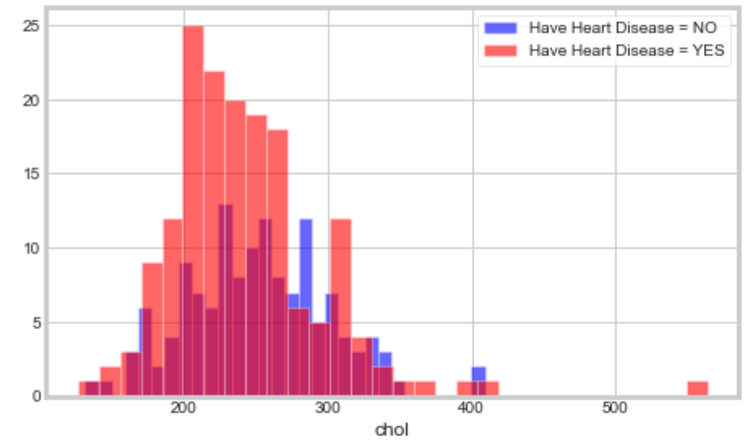
2) people with exang of 0 are more often have been diagonised with the heart disease

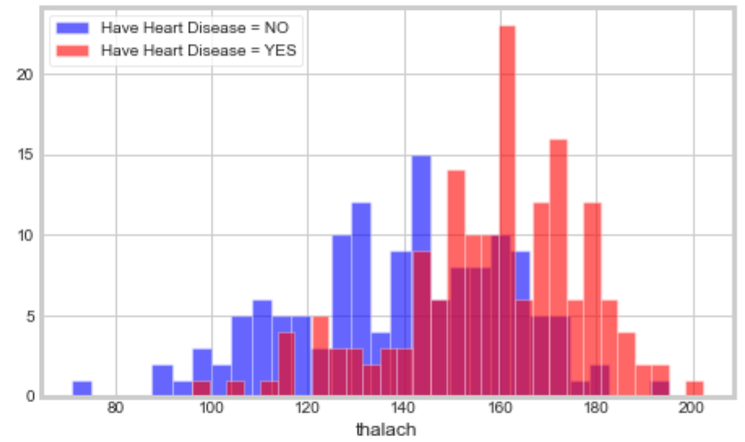
3) people with fbs of 120 and more are more often have been diagonised with the heart disease

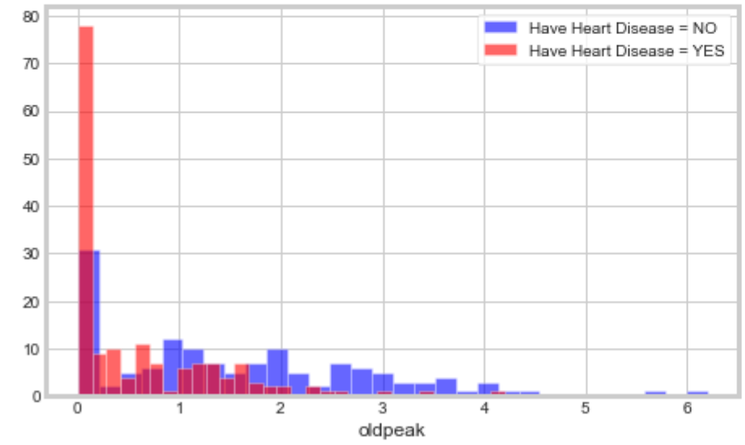
In [372]:
```
plt.figure(figsize=(15,15))
for i,column in enumerate(df_num,1):
    plt.subplot(3,2,i)
    df[df['target']==0][column].hist(bins=30,color='blue',label='Have Heart Disease = NO',alpha=0.6)
    df[df['target']==1][column].hist(bins=30,color='red',label='Have Heart Disease = YES',alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```











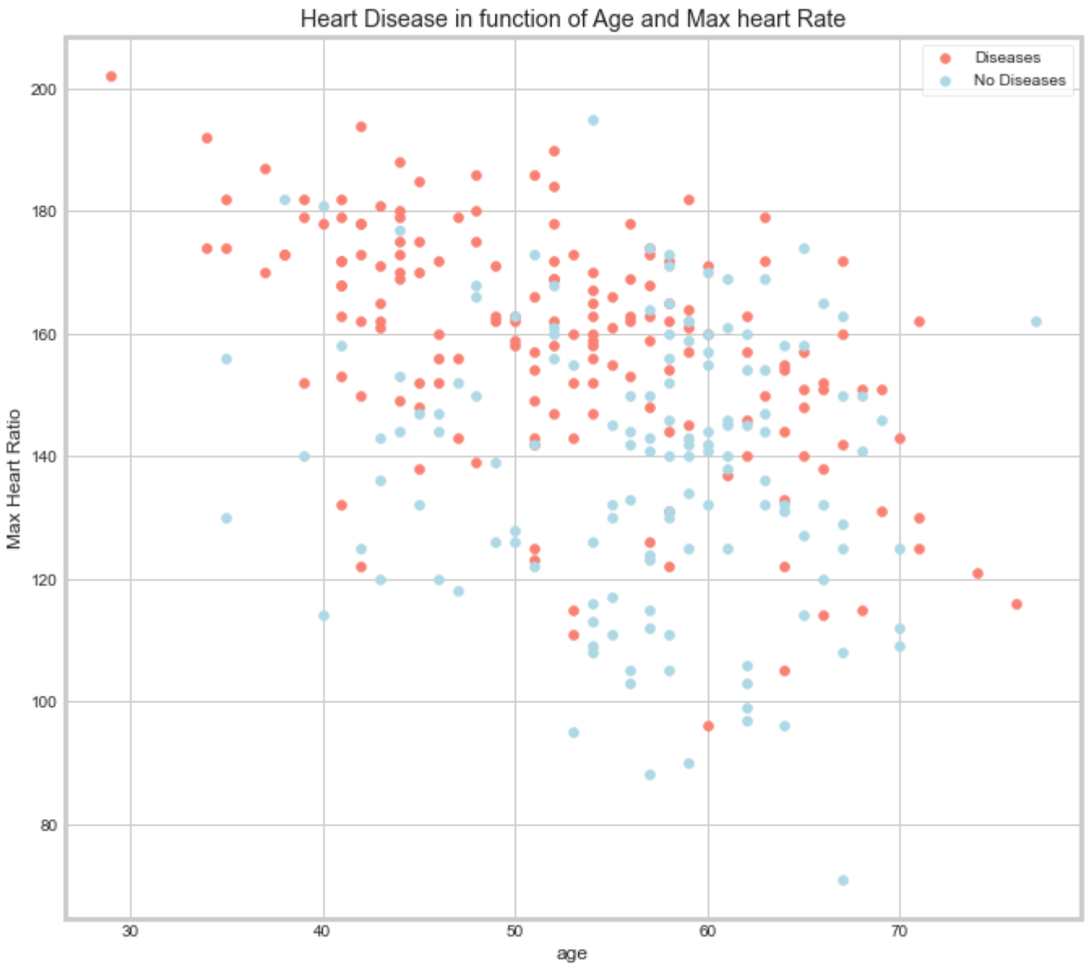
Major Insights Drawn from Above:

1) cholestrol level above 200 is a greater cause for heart disease

2) blood pressure(trestbps) 130-140 are a greater concern for the heart diesease

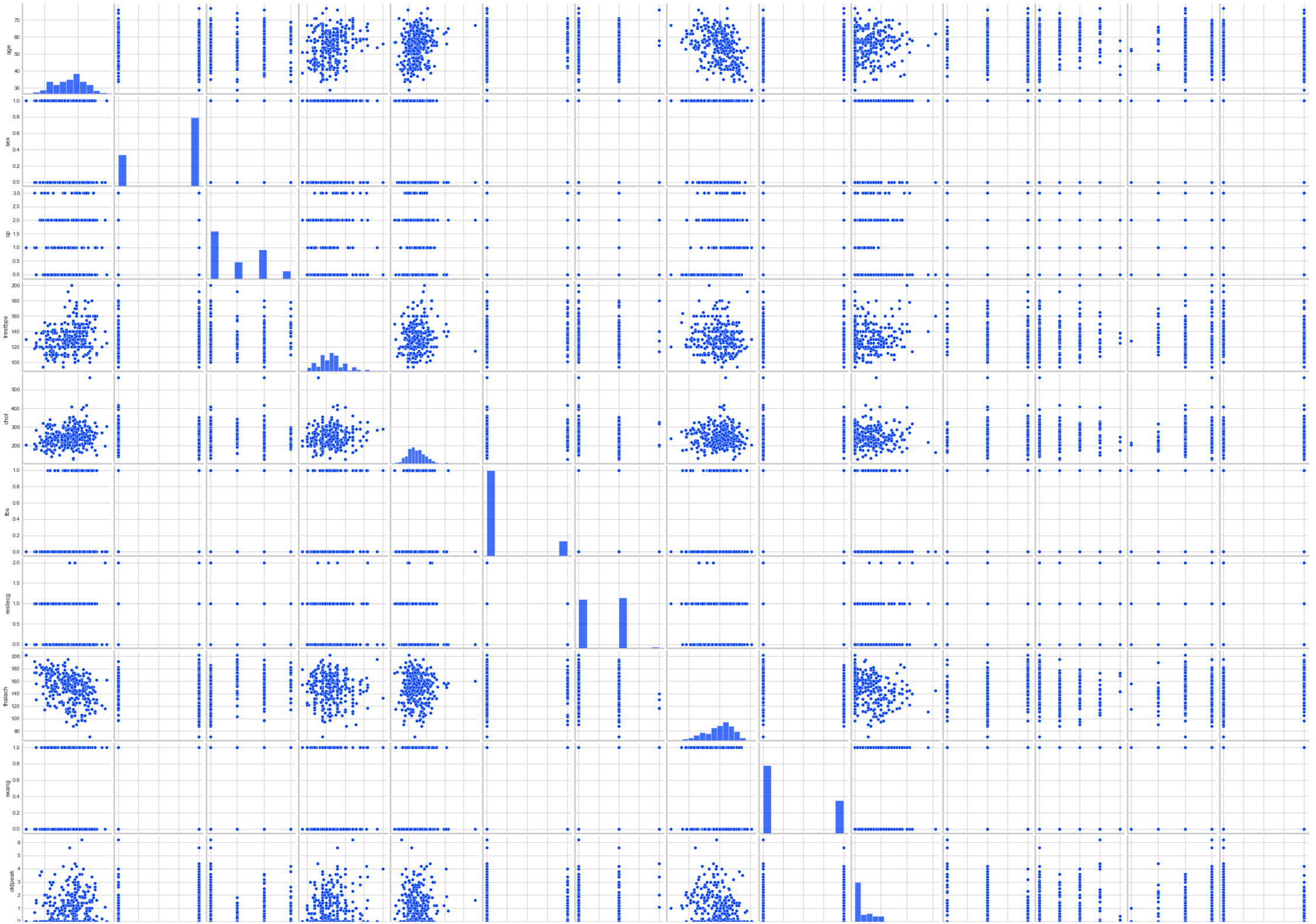
3) thalach(max heart rate) above 140 is a greater cause for heart disease

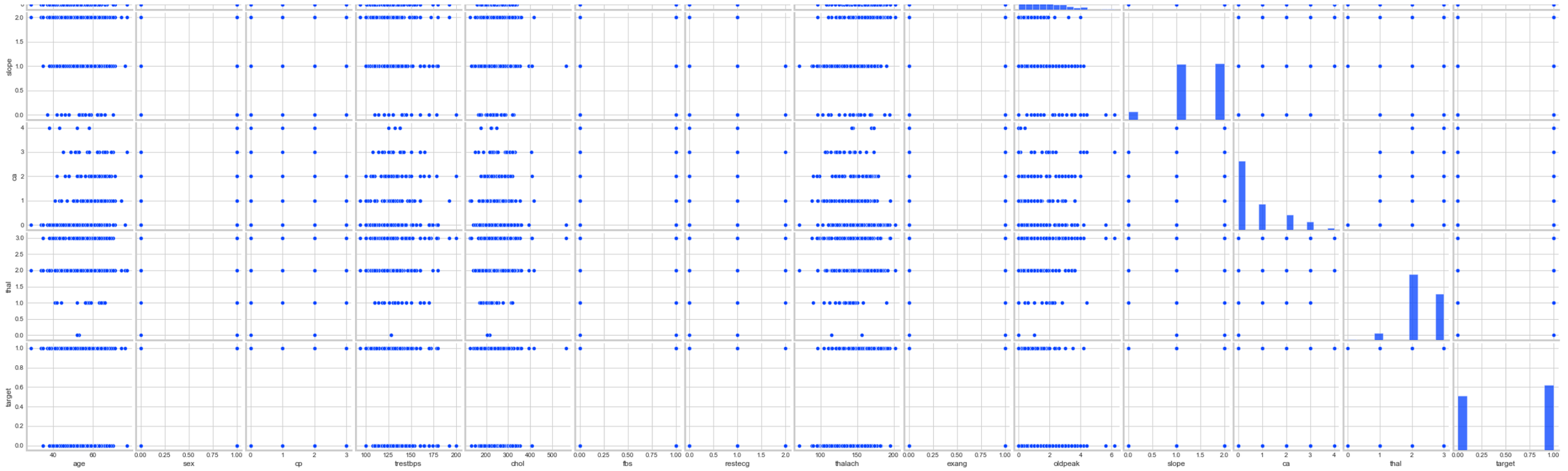
In [484]:
```
plt.figure(figsize=(10,10))

plt.scatter(df.age[df['target']==1],df.thalach[df['target']==1],c='salmon')

plt.scatter(df.age[df['target']==0],df.thalach[df['target']==0],c='lightblue')

plt.title("Heart Disease in function of Age and Max heart Rate")

plt.xlabel("age")

plt.ylabel('Max Heart Ratio')

plt.legend(['Diseases','No Diseases']);
```



In [467]:
```
sns.pairplot(df)
```

Out[467]:
```
<seaborn.axisgrid.PairGrid at 0x24d03ab15b0>
```





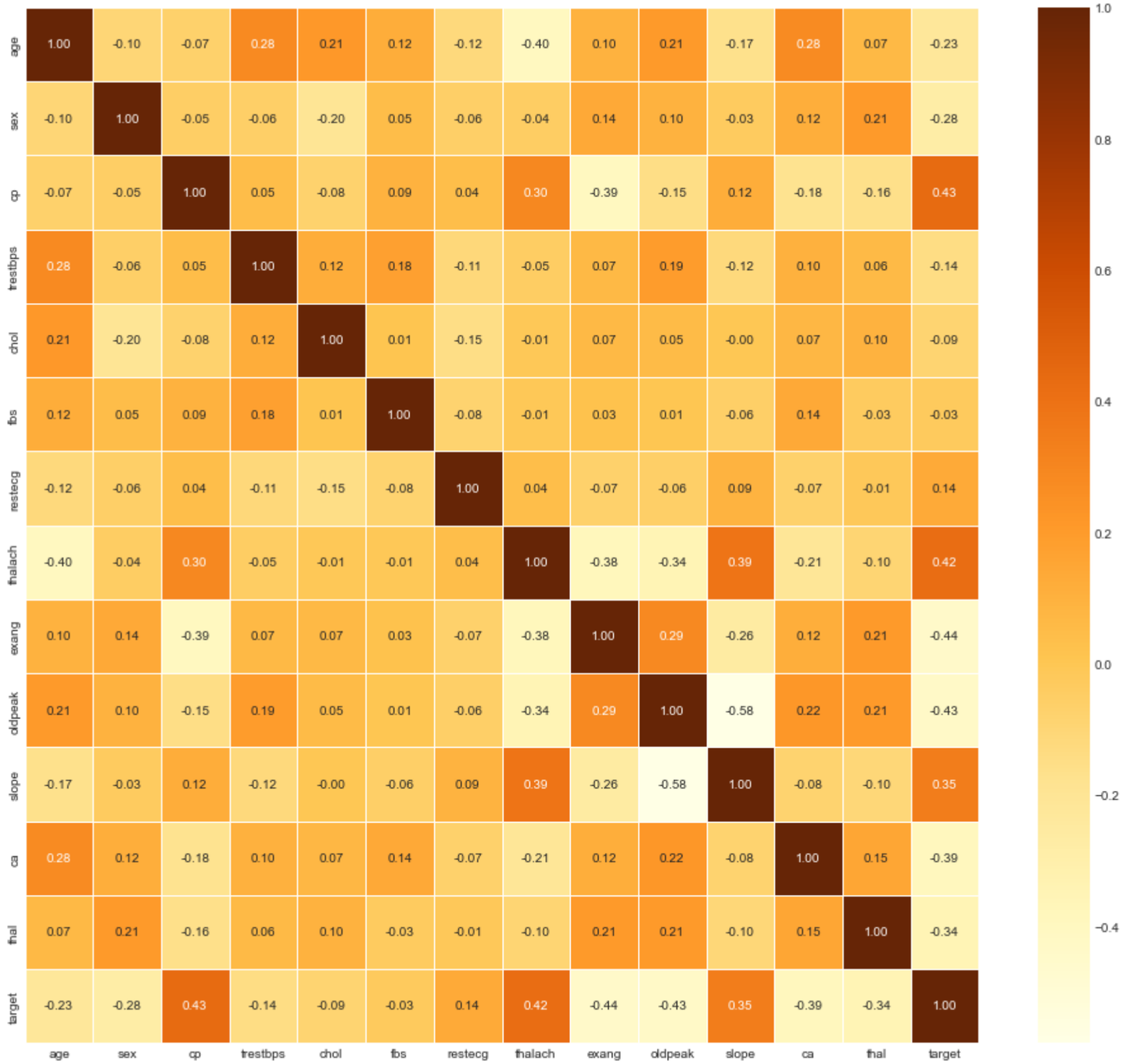
In [468]:
```
corr=df.corr()
```

In [375]:
```
corr
```

Out[375]:

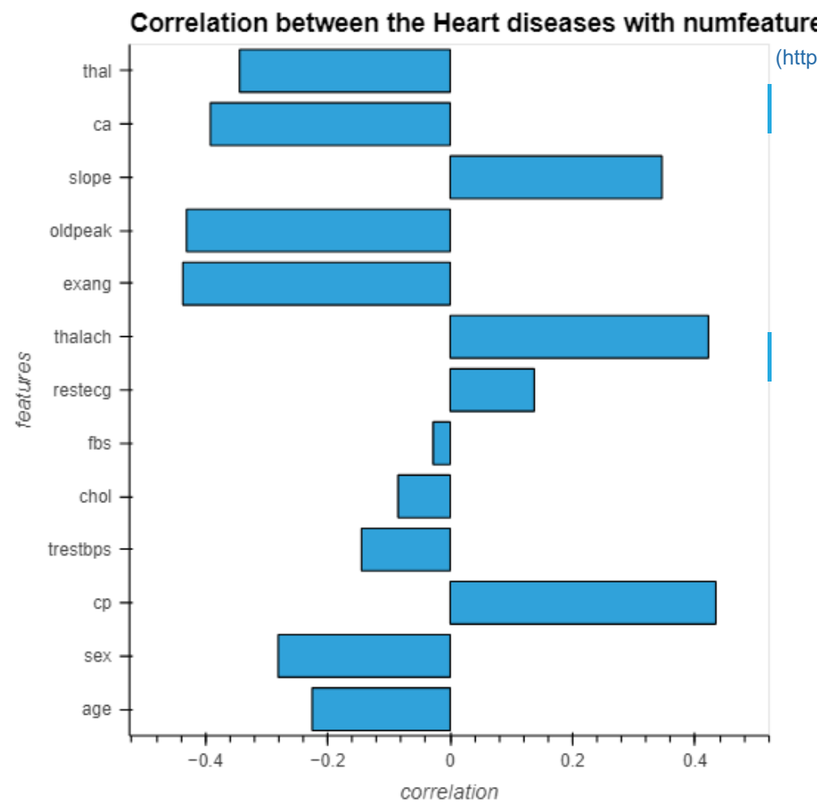
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276326	0.068001	-0.225439
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118261	0.210041	-0.280937
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717	-0.181053	-0.161736	0.433798
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389	0.062210	-0.144931
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511	0.098803	-0.085239
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979	-0.032019	-0.028046
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042	-0.011981	0.137230
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177	-0.096439	0.421741
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739	0.206754	-0.436757
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682	0.210244	-0.430696
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155	-0.104764	0.345877
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000	0.151832	-0.391724
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832	1.000000	-0.344029
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1.000000

In [376]:
```
fig,ax = plt.subplots(figsize=(15,15))
ax= sns.heatmap(corr,annot=True,linewidths=0.5,fmt=".2f",cmap='YlOrBr')
```



In [377]:
```
df.drop('target',axis=1).corrwith(df['target']).hvplot.barh(width=500,height=500,
title="Correlation between the Heart diseases with numfeatures",xlabel='features',ylabel='correlation')
```

Out[377]:



fbs,chol are the lowest correlated features with the target variable

In [378]:
```
df_cat
df_cat.remove('target')
```

In [379]:
```
df_cat_dummies = pd.get_dummies(df[df_cat], columns = df_cat)
```

In [380]:
```
df.oldpeak.dtype
```

Out[380]:
```
dtype('float64')
```

In [381]:
```
df_cat_dummies
```

Out[381]:

	sex_0	sex_1	cp_0	cp_1	cp_2	cp_3	fbs_0	fbs_1	restecg_0	restecg_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	0	1	0	0	0	1	0	1	1	0	...	0	1	0	0	0	0	0	1	0	0
1	0	1	0	0	1	0	1	0	0	1	...	0	1	0	0	0	0	0	0	1	0
2	1	0	0	1	0	0	1	0	1	0	...	1	1	0	0	0	0	0	0	1	0
3	0	1	0	1	0	0	1	0	0	1	...	1	1	0	0	0	0	0	0	1	0
4	1	0	1	0	0	0	1	0	0	1	...	1	1	0	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	1	0	1	0	0	0	1	0	0	1	...	0	1	0	0	0	0	0	0	0	1
299	0	1	0	0	0	1	1	0	0	1	...	0	1	0	0	0	0	0	0	0	1
300	0	1	1	0	0	0	0	1	0	1	...	0	0	0	1	0	0	0	0	0	1
301	0	1	1	0	0	0	1	0	0	1	...	0	0	1	0	0	0	0	0	0	1
302	1	0	0	1	0	0	1	0	1	0	...	0	0	1	0	0	0	0	0	1	0

303 rows × 25 columns

In [382]:
```
df_cat_dummies.shape
```

Out[382]: (303, 25)

In [383]:
```
df_cat_dummies.head(35).T
```

Out[383]:

	0	1	2	3	4	5	6	7	8	9	...	25	26	27	28	29	30	31	32	33	34
sex_0	0	0	1	0	1	0	1	0	0	0	...	1	0	0	1	0	1	0	0	0	0
sex_1	1	1	0	1	0	1	0	1	1	1	...	0	1	1	0	1	0	1	1	1	1
cp_0	0	0	0	0	1	1	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
cp_1	0	0	1	1	0	0	1	1	0	0	...	1	0	0	0	0	1	0	1	0	0
cp_2	0	1	0	0	0	0	0	0	1	1	...	0	1	1	1	1	0	0	0	1	0
cp_3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
fbs_0	0	1	1	1	1	1	1	1	0	1	...	1	0	1	0	0	1	1	1	1	1
fbs_1	1	0	0	0	0	0	0	0	1	0	...	0	1	0	1	1	0	0	0	0	0
restecg_0	1	0	1	0	0	0	1	0	0	0	...	0	0	0	1	1	0	0	1	1	1
restecg_1	0	1	0	1	1	1	0	1	1	1	...	1	1	1	0	0	1	1	0	0	0
restecg_2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
exang_0	1	1	1	1	0	1	1	1	1	1	...	1	1	1	1	1	1	1	1	1	0
exang_1	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
slope_0	1	1	0	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	1	0
slope_1	0	0	0	0	0	1	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
slope_2	0	0	1	1	1	0	0	1	1	1	...	1	1	1	1	0	1	1	1	0	1
ca_0	1	1	1	1	1	1	1	1	1	1	...	0	1	1	0	1	0	1	1	0	0
ca_1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	1	0	0	1	1
ca_2	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0
ca_3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
ca_4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
thal_0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
thal_1	1	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
thal_2	0	1	1	1	1	0	1	0	0	1	...	1	1	1	1	1	1	0	1	1	1
thal_3	0	0	0	0	0	0	0	1	1	0	...	0	0	0	0	0	0	1	0	0	0

25 rows × 35 columns

In [384]:
```
from sklearn.preprocessing import StandardScaler
```

In [385]:
```
sc = StandardScaler()
```

In [386]:
```
df_num_cols = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

In [387]:
```
df_scaled = sc.fit_transform(df[df_num_cols])
```

In [388]:
```
df_scaled
```

Out[388]:
```
array([[ 0.9521966 ,  0.76395577, -0.25633371,  0.01544279,  1.08733806],
       [-1.91531289, -0.09273778,  0.07219949,  1.63347147,  2.12257273],
       [-1.47415758, -0.09273778, -0.81677269,  0.97751389,  0.31091206],
       ...,
       [ 1.50364073,  0.70684287, -1.029353  , -0.37813176,  2.03630317],
       [ 0.29046364, -0.09273778, -2.2275329 , -1.51512489,  0.13837295],
       [ 0.29046364, -0.09273778, -0.19835726,  1.0649749 , -0.89686172]])
```

In [389]:
```
df_scaled_labeled = pd.DataFrame(df_scaled,columns=df_num_cols)
```

In [390]:
```
df_scaled_labeled
```

Out[390]:

	age	trestbps	chol	thalach	oldpeak
0	0.952197	0.763956	-0.256334	0.015443	1.087338
1	-1.915313	-0.092738	0.072199	1.633471	2.122573
2	-1.474158	-0.092738	-0.816773	0.977514	0.310912
3	0.180175	-0.663867	-0.198357	1.239897	-0.206705
4	0.290464	-0.663867	2.082050	0.583939	-0.379244
...	...	...	...	...	...
298	0.290464	0.478391	-0.101730	-1.165281	-0.724323
299	-1.033002	-1.234996	0.342756	-0.771706	0.138373
300	1.503641	0.706843	-1.029353	-0.378132	2.036303
301	0.290464	-0.092738	-2.227533	-1.515125	0.138373
302	0.290464	-0.092738	-0.198357	1.064975	-0.896862

303 rows × 5 columns

In [391]:
```
df_final = pd.concat([df_cat_dummies,df_scaled_labeled],axis=1)
```

In [392]:
```
df_final=df_final.drop(df_final.iloc[:,0:4],axis=1)
```

In [393]:
```
df_final.head().T
```

Out[393]:

	0	1	2	3	4
cp_2	0.000000	1.000000	0.000000	0.000000	0.000000
cp_3	1.000000	0.000000	0.000000	0.000000	0.000000
fbs_0	0.000000	1.000000	1.000000	1.000000	1.000000
fbs_1	1.000000	0.000000	0.000000	0.000000	0.000000
restecg_0	1.000000	0.000000	1.000000	0.000000	0.000000
restecg_1	0.000000	1.000000	0.000000	1.000000	1.000000
restecg_2	0.000000	0.000000	0.000000	0.000000	0.000000
exang_0	1.000000	1.000000	1.000000	1.000000	0.000000
exang_1	0.000000	0.000000	0.000000	0.000000	1.000000
slope_0	1.000000	1.000000	0.000000	0.000000	0.000000
slope_1	0.000000	0.000000	0.000000	0.000000	0.000000
slope_2	0.000000	0.000000	1.000000	1.000000	1.000000
ca_0	1.000000	1.000000	1.000000	1.000000	1.000000
ca_1	0.000000	0.000000	0.000000	0.000000	0.000000
ca_2	0.000000	0.000000	0.000000	0.000000	0.000000
ca_3	0.000000	0.000000	0.000000	0.000000	0.000000
ca_4	0.000000	0.000000	0.000000	0.000000	0.000000
thal_0	0.000000	0.000000	0.000000	0.000000	0.000000
thal_1	1.000000	0.000000	0.000000	0.000000	0.000000
thal_2	0.000000	1.000000	1.000000	1.000000	1.000000
thal_3	0.000000	0.000000	0.000000	0.000000	0.000000
age	0.952197	-1.915313	-1.474158	0.180175	0.290464
trestbps	0.763956	-0.092738	-0.092738	-0.663867	-0.663867
chol	-0.256334	0.072199	-0.816773	-0.198357	2.082050
thalach	0.015443	1.633471	0.977514	1.239897	0.583939
oldpeak	1.087338	2.122573	0.310912	-0.206705	-0.379244

Building a basemodel with all the independent features

In [394]:
```
X = df_final
y = df['target']
```

In [395]:
```
from sklearn.model_selection import train_test_split
```

In [396]:
```
X_train, X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42,stratify=y)
```

In [397]:
```
X_train.shape, X_test.shape,y_train.shape,y_test.shape
```

Out[397]:
```
((242, 26), (61, 26), (242,), (61,))
```

In [398]:
```
from sklearn.linear_model import LogisticRegression

lr= LogisticRegression()
lr.fit(X_train,y_train)
```

Out[398]:
```
LogisticRegression()
```

In [399]:
```
prediction_1 = lr.predict(X_test)
```

In [400]:
```
prediction_1
```

Out[400]:
```
array([0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1], dtype=int64)
```

In [401]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [456]:
```
WithoutFS=accuracy_score(y_test,prediction_1)
```

In [457]:
```
WithoutFS
```

Out[457]:
```
0.819672131147541
```

In [403]:
```
cm1 = confusion_matrix(y_test,prediction_1)
```

In [404]:
```
cm1
```

Out[404]:
```
array([[19,  9],
       [ 2, 31]], dtype=int64)
```

In [405]:
```
clf_report1 = classification_report(y_test,prediction_1)
```

In [406]:
```
clf_report=pd.DataFrame(classification_report(y_test,prediction_1,output_dict=True))
```

In [407]:
```
clf_report
```

Out[407]:

	0	1	accuracy	macro avg	weighted avg
precision	0.904762	0.775000	0.819672	0.839881	0.834563
recall	0.678571	0.939394	0.819672	0.808983	0.819672
f1-score	0.775510	0.849315	0.819672	0.812413	0.815437
support	28.000000	33.000000	0.819672	61.000000	61.000000

In [408]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=1000,n_jobs=-1)
rf.fit(X_train,y_train)
prediction_2 = rf.predict(X_test)
```

In [409]:
```
prediction_2
```

Out[409]:
```
array([0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1], dtype=int64)
```

In [410]:
```
accuracy_score(y_test,prediction_2)
```

Out[410]:
```
0.7868852459016393
```

In [411]:
```
cm2 = confusion_matrix(y_test,prediction_2)
```

In [412]:
```
cm2
```

Out[412]:
```
array([[19,  9],
       [ 4, 29]], dtype=int64)
```

In [413]:
```
df_final.columns
```

Out[413]:
```
Index(['cp_2', 'cp_3', 'fbs_0', 'fbs_1', 'restecg_0', 'restecg_1', 'restecg_2',
       'exang_0', 'exang_1', 'slope_0', 'slope_1', 'slope_2', 'ca_0', 'ca_1',
       'ca_2', 'ca_3', 'ca_4', 'thal_0', 'thal_1', 'thal_2', 'thal_3', 'age',
       'trestbps', 'chol', 'thalach', 'oldpeak'],
      dtype='object')
```

Building a model with the implementing the feature selection methods

In [414]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

In [415]:
```
from statsmodels.tools import add_constant as ac
df_final_constant = ac(df)
df_final_constant.head()
```

Out[415]:

	const	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	1.0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	1.0	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	1.0	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	1.0	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	1.0	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Feature Selection using the stats model

In [416]:
```
from scipy import stats as st
```

In [417]:
```
st.chisqprob = lambda chisq,df: st.chi2.smf(chisq,df)
cols = df_final_constant.columns[:-1]
model = sm.Logit(df.target,df_final_constant[cols])
result = model.fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.348904
         Iterations 7
```

Out[417]:

Logit Regression Results

Dep. Variable:	target	**No. Observations:**	303
Model:	Logit	**Df Residuals:**	289
Method:	MLE	**Df Model:**	13
Date:	Sat, 06 Aug 2022	**Pseudo R-squ.:**	0.4937
Time:	09:06:47	**Log-Likelihood:**	-105.72
converged:	True	**LL-Null:**	-208.82
Covariance Type:	nonrobust	**LLR p-value:**	7.262e-37

	coef	std err	z	P>|z|	[0.025	0.975]
const	3.4505	2.571	1.342	0.180	-1.590	8.490
age	-0.0049	0.023	-0.212	0.832	-0.050	0.041
sex	-1.7582	0.469	-3.751	0.000	-2.677	-0.839
cp	0.8599	0.185	4.638	0.000	0.496	1.223
trestbps	-0.0195	0.010	-1.884	0.060	-0.040	0.001
chol	-0.0046	0.004	-1.224	0.221	-0.012	0.003
fbs	0.0349	0.529	0.066	0.947	-1.003	1.073
restecg	0.4663	0.348	1.339	0.181	-0.216	1.149
thalach	0.0232	0.010	2.219	0.026	0.003	0.044
exang	-0.9800	0.410	-2.391	0.017	-1.783	-0.177
oldpeak	-0.5403	0.214	-2.526	0.012	-0.959	-0.121
slope	0.5793	0.350	1.656	0.098	-0.106	1.265
ca	-0.7733	0.191	-4.051	0.000	-1.147	-0.399
thal	-0.9004	0.290	-3.104	0.002	-1.469	-0.332

In [419]:
```
def bfe(df,depvar,col_list):
    while len(col_list)>0:
        model =sm.Logit(depvar,df[col_list])
        result=model.fit(disp=0)
        largest_pvalue=round(result.pvalues,3).nlargest(1)
        if largest_pvalue[0]<(0.05):
            return result
        else:
            col_list=col_list.drop(largest_pvalue.index)
result= bfe(df_final_constant,df.target,cols)
```

In [420]:
```
result.summary()
```

Out[420]:

Logit Regression Results

Dep. Variable:	target	**No. Observations:**	303
Model:	Logit	**Df Residuals:**	296
Method:	MLE	**Df Model:**	6
Date:	Sat, 06 Aug 2022	**Pseudo R-squ.:**	0.4651
Time:	09:08:33	**Log-Likelihood:**	-111.71
converged:	True	**LL-Null:**	-208.82
Covariance Type:	nonrobust	**LLR p-value:**	3.209e-39

	coef	std err	z	P>|z|	[0.025	0.975]
sex	-1.3898	0.405	-3.431	0.001	-2.184	-0.596
cp	0.7861	0.174	4.509	0.000	0.444	1.128
thalach	0.0261	0.004	5.905	0.000	0.017	0.035
exang	-1.0130	0.376	-2.695	0.007	-1.750	-0.276
oldpeak	-0.7262	0.176	-4.130	0.000	-1.071	-0.382
ca	-0.7053	0.173	-4.087	0.000	-1.043	-0.367
thal	-0.8674	0.259	-3.351	0.001	-1.375	-0.360

Based on the logit model the 'sex','cp','thalach','exang','oldpeak','ca','thal' are the features good for the model fitting as p value is less than the significance level of 5%

In [422]:
```
X1 = df_final_constant[['sex','cp','thalach','exang','oldpeak','ca','thal']]
y1 = df['target']
```

In [424]:
```
df_cat1 = []
df_num1=[]
for column in df_final_constant.columns:
    if len(df_final_constant[column].unique())<=10:
        df_cat1.append(column)
    else:
        df_num1.append(column)
```

In [425]:
```
df_cat1
```

Out[425]:
```
['const',
 'sex',
 'cp',
 'fbs',
 'restecg',
 'exang',
 'slope',
 'ca',
 'thal',
 'target']
```

In [427]:
```
df_num1
```

Out[427]:
```
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

In [428]:
```
df_cat1.remove('target')
```

In [431]:
```
df_dummies1 = pd.get_dummies(df_final_constant[df_cat1],columns=df_cat1)
```

In [432]:
```
df_dummies1
```

Out[432]:

	const_1.0	sex_0	sex_1	cp_0	cp_1	cp_2	cp_3	fbs_0	fbs_1	restecg_0	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	1	0	1	0	0	0	1	0	1	1	...	0	1	0	0	0	0	0	1	0	0
1	1	0	1	0	0	1	0	1	0	0	...	0	1	0	0	0	0	0	0	1	0
2	1	1	0	0	1	0	0	1	0	1	...	1	1	0	0	0	0	0	0	1	0
3	1	0	1	0	1	0	0	1	0	0	...	1	1	0	0	0	0	0	0	1	0
4	1	1	0	1	0	0	0	1	0	0	...	1	1	0	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	1	1	0	1	0	0	0	1	0	0	...	0	1	0	0	0	0	0	0	0	1
299	1	0	1	0	0	0	1	1	0	0	...	0	1	0	0	0	0	0	0	0	1
300	1	0	1	1	0	0	0	0	1	0	...	0	0	0	1	0	0	0	0	0	1
301	1	0	1	1	0	0	0	1	0	0	...	0	0	1	0	0	0	0	0	0	1
302	1	1	0	0	1	0	0	1	0	1	...	0	0	1	0	0	0	0	0	1	0

303 rows × 26 columns

In [433]:
```
df_num1scaled = sc.fit_transform(df_final_constant[df_num1])
```

In [434]:
```
df_num1scaled
```

Out[434]:
```
array([[ 0.9521966 ,  0.76395577, -0.25633371,  0.01544279,  1.08733806],
       [-1.91531289, -0.09273778,  0.07219949,  1.63347147,  2.12257273],
       [-1.47415758, -0.09273778, -0.81677269,  0.97751389,  0.31091206],
       ...,
       [ 1.50364073,  0.70684287, -1.029353  , -0.37813176,  2.03630317],
       [ 0.29046364, -0.09273778, -2.2275329 , -1.51512489,  0.13837295],
       [ 0.29046364, -0.09273778, -0.19835726,  1.0649749 , -0.89686172]])
```

In [435]:
```
df_num1scaled=pd.DataFrame(df_num1scaled,columns=df_num1)
```

In [436]:
```
df_num1scaled
```

Out[436]:

	age	trestbps	chol	thalach	oldpeak
0	0.952197	0.763956	-0.256334	0.015443	1.087338
1	-1.915313	-0.092738	0.072199	1.633471	2.122573
2	-1.474158	-0.092738	-0.816773	0.977514	0.310912
3	0.180175	-0.663867	-0.198357	1.239897	-0.206705
4	0.290464	-0.663867	2.082050	0.583939	-0.379244
...	...	...	...	...	...
298	0.290464	0.478391	-0.101730	-1.165281	-0.724323
299	-1.033002	-1.234996	0.342756	-0.771706	0.138373
300	1.503641	0.706843	-1.029353	-0.378132	2.036303
301	0.290464	-0.092738	-2.227533	-1.515125	0.138373
302	0.290464	-0.092738	-0.198357	1.064975	-0.896862

303 rows × 5 columns

In [437]:
```
df_final2 = pd.concat([df_num1scaled,df_dummies1],axis=1)
```

In [438]:
```
df_final2.head()
```

Out[438]:

	age	trestbps	chol	thalach	oldpeak	const_1.0	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	0.952197	0.763956	-0.256334	0.015443	1.087338	1	0	1	0	0	...	0	1	0	0	0	0	0	1	0	0
1	-1.915313	-0.092738	0.072199	1.633471	2.122573	1	0	1	0	0	...	0	1	0	0	0	0	0	0	1	0
2	-1.474158	-0.092738	-0.816773	0.977514	0.310912	1	1	0	0	1	...	1	1	0	0	0	0	0	0	1	0
3	0.180175	-0.663867	-0.198357	1.239897	-0.206705	1	0	1	0	1	...	1	1	0	0	0	0	0	0	1	0
4	0.290464	-0.663867	2.082050	0.583939	-0.379244	1	1	0	1	0	...	1	1	0	0	0	0	0	0	1	0

5 rows × 31 columns

In [440]:
```
X1 = df_final2
y1 = df['target']
```

In [441]:
```
X1_train,X1_test,y1_train,y1_test = train_test_split(X1,y1,test_size=0.2,random_state=42,stratify=y)
```

In [442]:
```
X1_train.shape,X1_test.shape,y1_train.shape,y1_test.shape
```

Out[442]: ((242, 31), (61, 31), (242,), (61,))

In [443]:
```
lr1 = LogisticRegression()
```

In [444]:
```
lr1.fit(X1_train,y1_train)
```

Out[444]: LogisticRegression()

In [447]:
```
prediction_3= lr1.predict(X1_test)
```

In [458]:
```
WithFS = accuracy_score(y1_test,prediction_3)
```

In [459]:
```
WithFS
```

Out[459]:
```
0.8688524590163934
```

In [449]:
```
cm3 = confusion_matrix(y1_test,prediction_3)
```

In [450]:
```
cm3
```

Out[450]:
```
array([[23,  5],
       [ 3, 30]], dtype=int64)
```

In [451]:
```
clr_report3 = classification_report(y1_test,prediction_3)
```

In [452]:
```
clr_report3
```

Out[452]:
```
'              precision    recall  f1-score   support\n\n           0       0.88      0.82      0.85        28\n           1       0.86      0.91      0.88        33\n\n    accura
cy                           0.87        61\n   macro avg       0.87      0.87      0.87        61\nweighted avg       0.87      0.87      0.87        61\n'
```

The model accuracy without feature selection is 81.96%

The model accuracy without feature selection is 86.88%

In [ ]:
```

```