In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
data = pd.read_excel(r"C:\Users\Rahul\Desktop\data.xlsx"); data.head()
```

Out[2]:

	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 [3]:
```python
data.dtypes
```

Out[3]:
```
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 [4]:
```
data.isnull().sum()
```

Out[4]:
```
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 [5]:
```
data.shape
```

Out[5]:
```
(303, 14)
```

In [6]:
```
data.describe()
```

Out[6]:

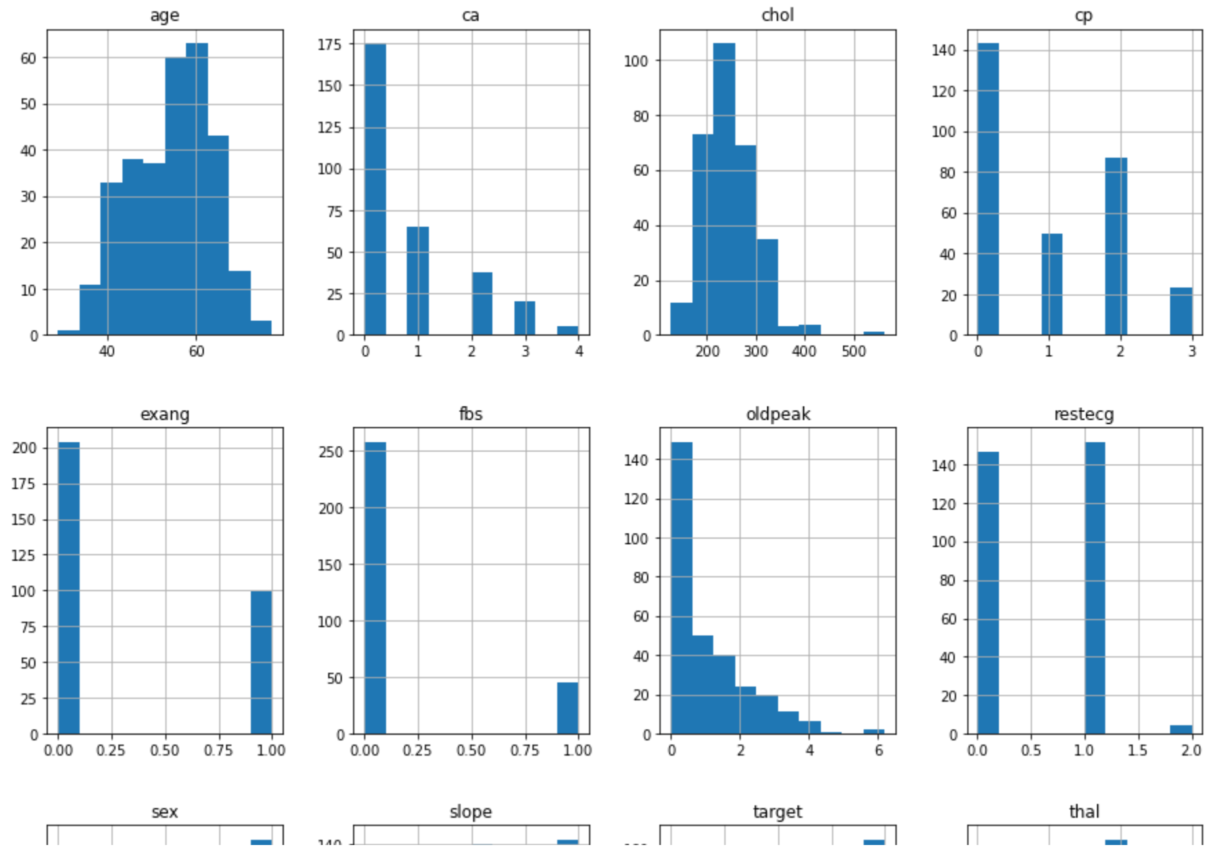
	age	sex	cp	trestbps	chol	fbs	restecg	t
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.(
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.(
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.(
min	29.000000	0.000000	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.(

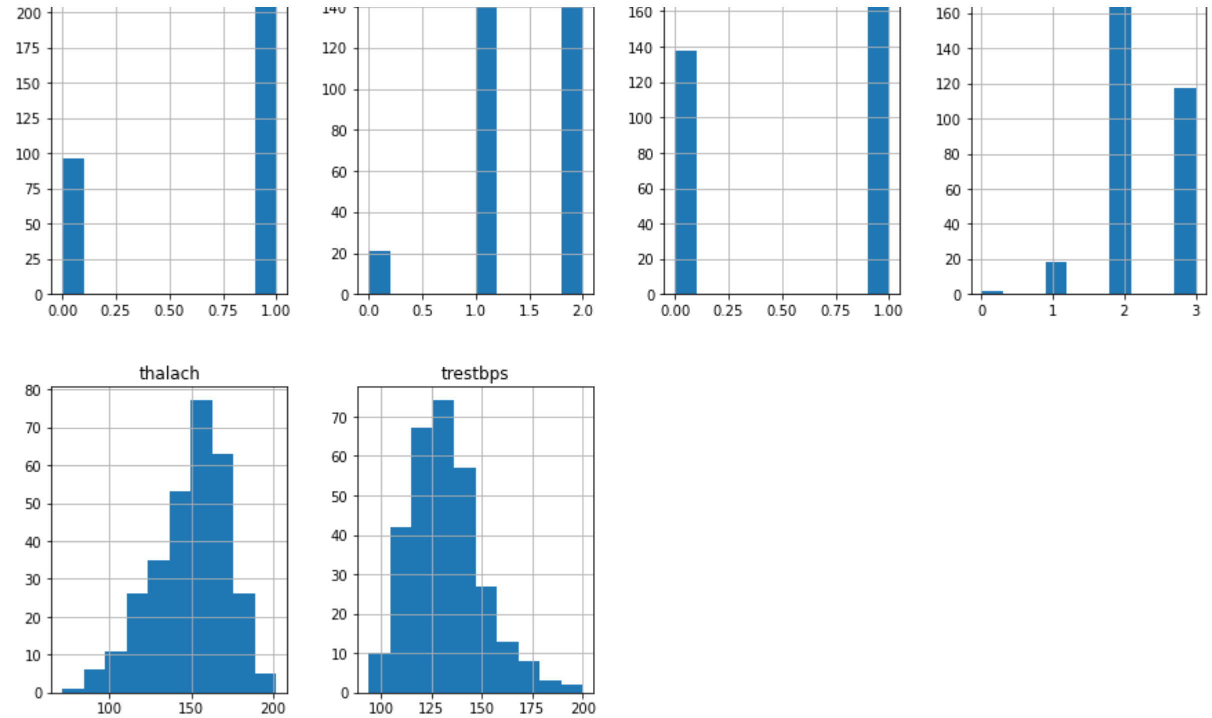
	age	sex	cp	trestbps	chol	fbs	restecg	t
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.(
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.(

Data Distribution

In [8]:

```
fig = plt.figure(figsize = (15,20))
ax = fig.gca()
data.hist(ax = ax)
plt.show()
```

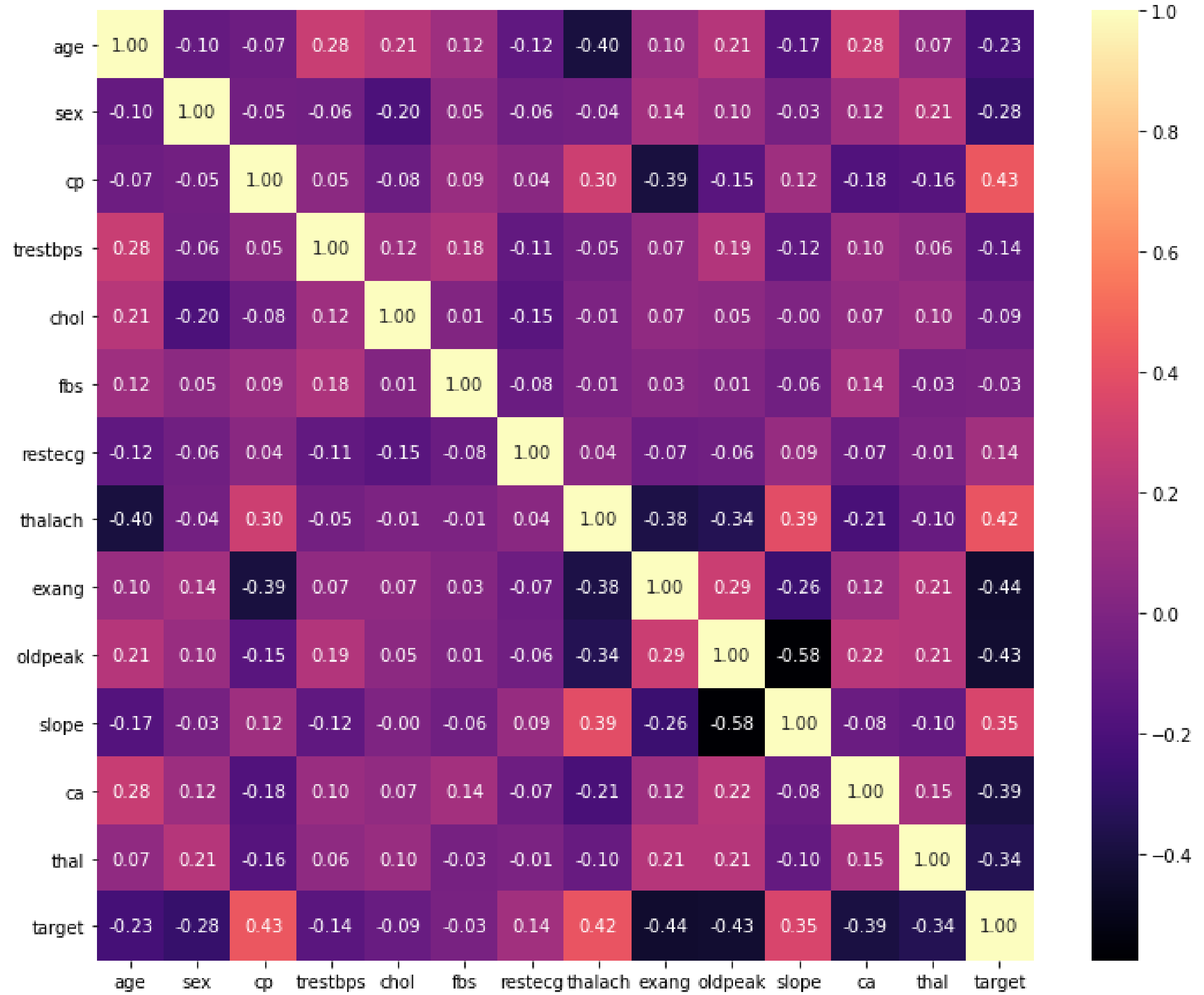




Data Analysis

In [9]:
```python
plt.figure(figsize=(12,10))
sns.heatmap(data.corr(),annot=True,cmap="magma",fmt='.2f')
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27816f7e20>
```



Heatmap makes it easy to classify the features are most relevant to the target variable, and we will plot the associated features of the heatmap using the seaborn library.Correlation shows whether the characteristics are related to each other or to the target variable. Correlation can be positive (increase in one value, the value of the objective variable increases) or negative (increase in one value, the value of the target variable decreased). From this heatmap we can observe that the ‘cp’ chest pain is highly related to the target variable. Compared to relation between other two variables we can say that chest pain contributes the most in prediction of presences of a heart disease.

Data Visualization

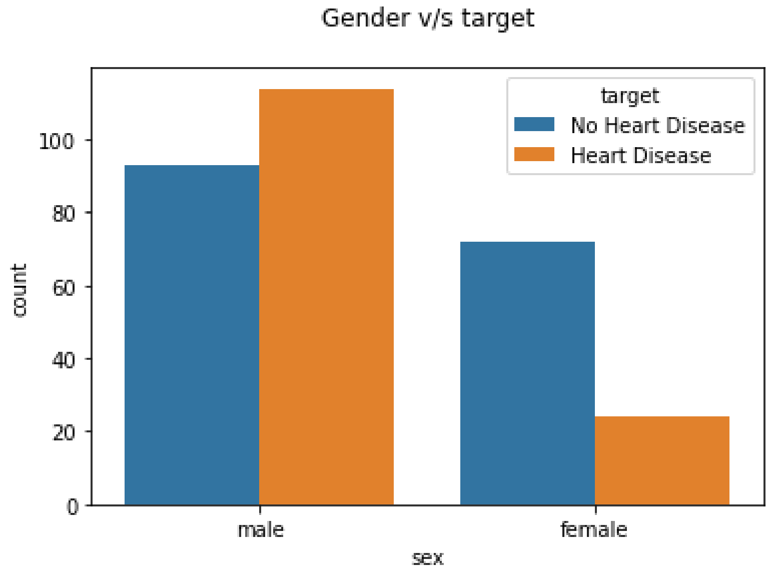
Countplot

In [10]:
```
df2 = data.copy()
def chng(sex):
    if sex == 0:
        return 'female'
    else:
        return 'male'

df2['sex'] = df2['sex'].apply(chng)
```

In [11]:
```
def chng2(target):
    if target == 0:
        return 'Heart Disease'
    else:
        return 'No Heart Disease'
```

In [12]:
```
df2['target'] = df2['target'].apply(chng2)
sns.countplot(data= df2, x='sex',hue='target')
plt.title('Gender v/s target\n')
```

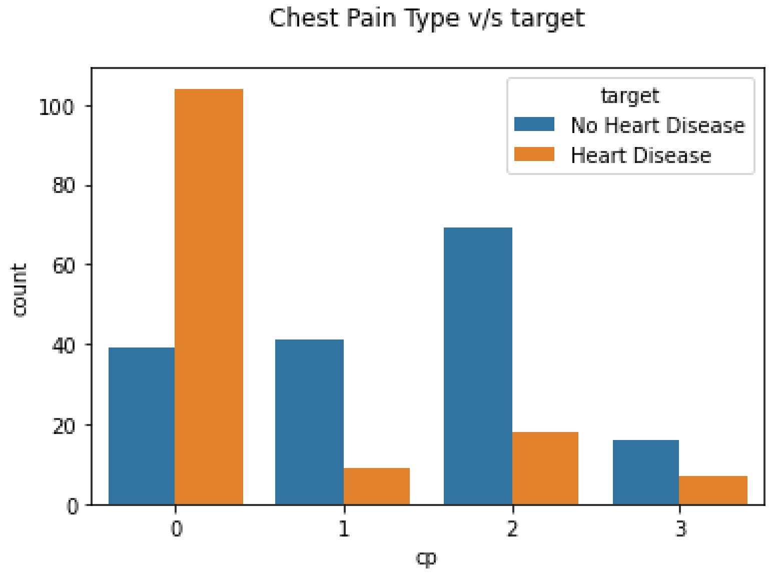
Out[12]:
```
Text(0.5, 1.0, 'Gender v/s target\n')
```



According to this dataset males are more susceptible to get Heart Disease than females. Men experience heart attacks more than women.

In [13]:
```
sns.countplot(data= df2, x='cp',hue='target')
plt.title('Chest Pain Type v/s target\n')
```

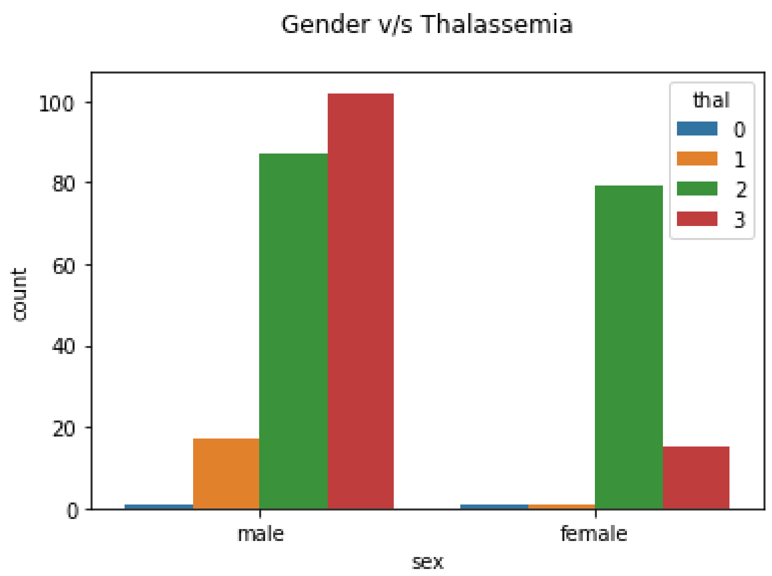
Out[13]:
```
Text(0.5, 1.0, 'Chest Pain Type v/s target\n')
```



There are four types of chest pain, asymptomatic, atypical angina, non-anginal pain and typical angina. Most of the Heart Disease patients are found to have asymptomatic chest pain.

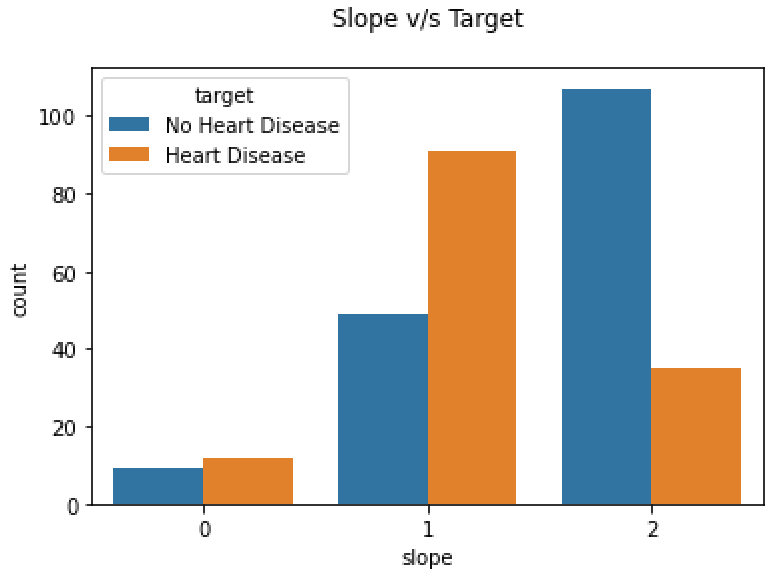
In [14]:
```
sns.countplot(data= df2, x='sex',hue='thal')
plt.title('Gender v/s Thalassemia\n')
print('Thalassemia (thal-uh-SEE-me-uh) is an inherited blood disorder t
hat causes your body to have less hemoglobin than normal. Hemoglobin en
ables red blood cells to carry oxygen')
```

```
Thalassemia (thal-uh-SEE-me-uh) is an inherited blood disorder that cau
ses your body to have less hemoglobin than normal. Hemoglobin enables r
ed blood cells to carry oxygen
```



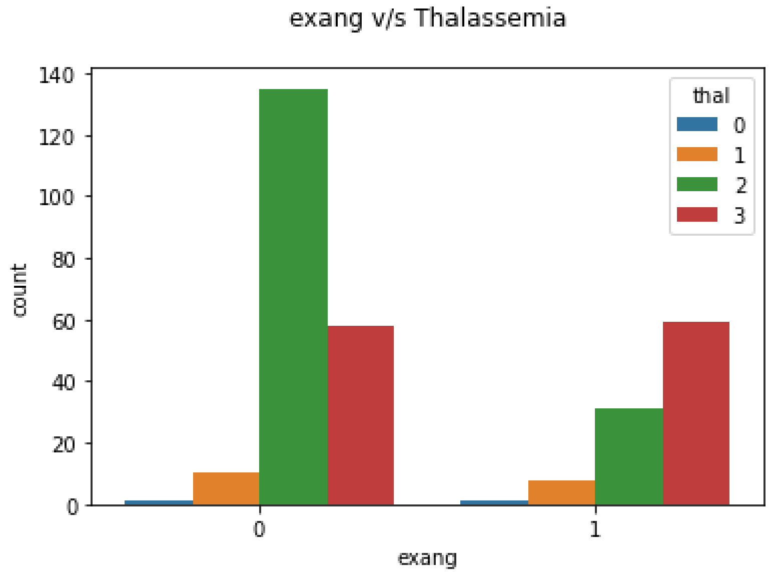
In [15]:
```
sns.countplot(data= df2, x='slope',hue='target')
plt.title('Slope v/s Target\n')
```

Out[15]: `Text(0.5, 1.0, 'Slope v/s Target\n')`

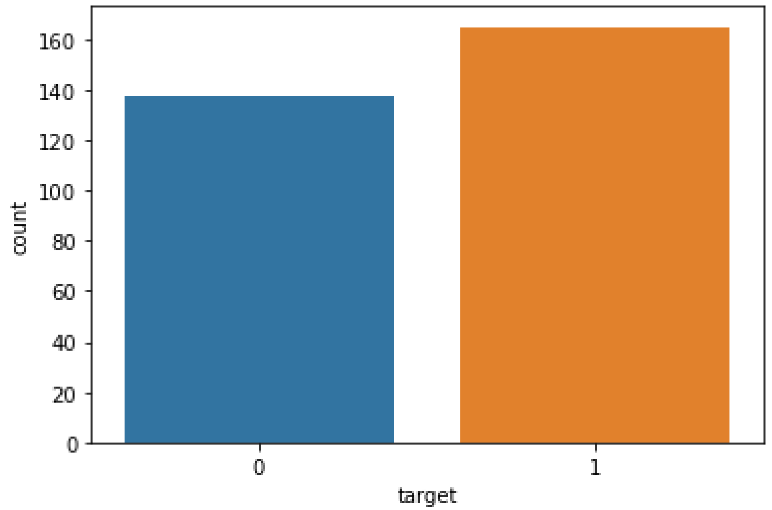


In [16]:
```
sns.countplot(data= df2, x='exang',hue='thal')
plt.title('exang v/s Thalassemia\n')
```

Out[16]:
```
Text(0.5, 1.0, 'exang v/s Thalassemia\n')
```



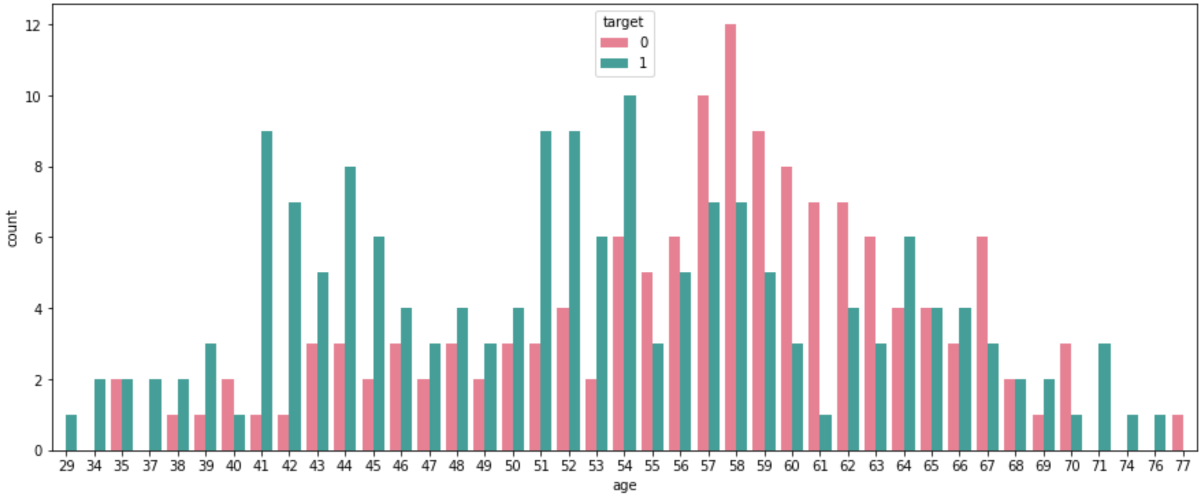
In [17]:
```
sns.countplot(x="target",data=data)
plt.show()
cases = data.target.value_counts()
print(f"There are {cases[0]} patients without heart disease and {cases[
1]} patients with the disease")
```



```
There are 138 patients without heart disease and 165 patients with the
disease
```

Number of people who have disease vs age¶

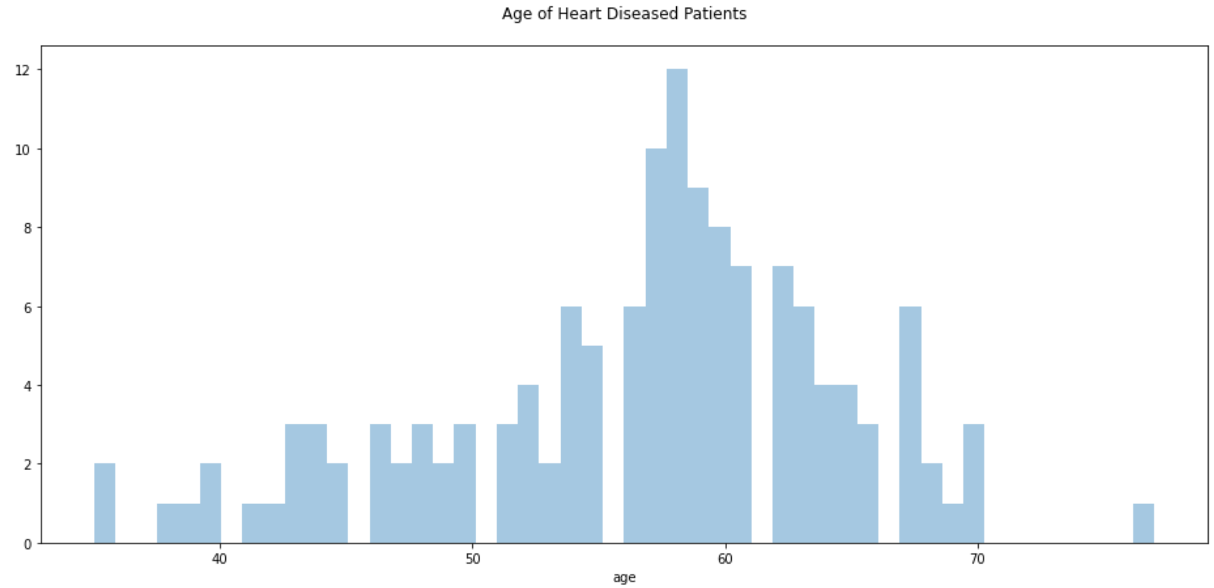
In [18]:
```python
plt.figure(figsize=(15,6))
sns.countplot(x='age',data = data, hue = 'target',palette='husl')
plt.show()
```



The people with the highest risk are between the ages of 41 and 54 i.e. the blue bars

In [19]:
```python
plt.figure(figsize=(16,7))
sns.distplot(data[data['target']==0]['age'],kde=False,bins=50)
plt.title('Age of Heart Diseased Patients\n')
```

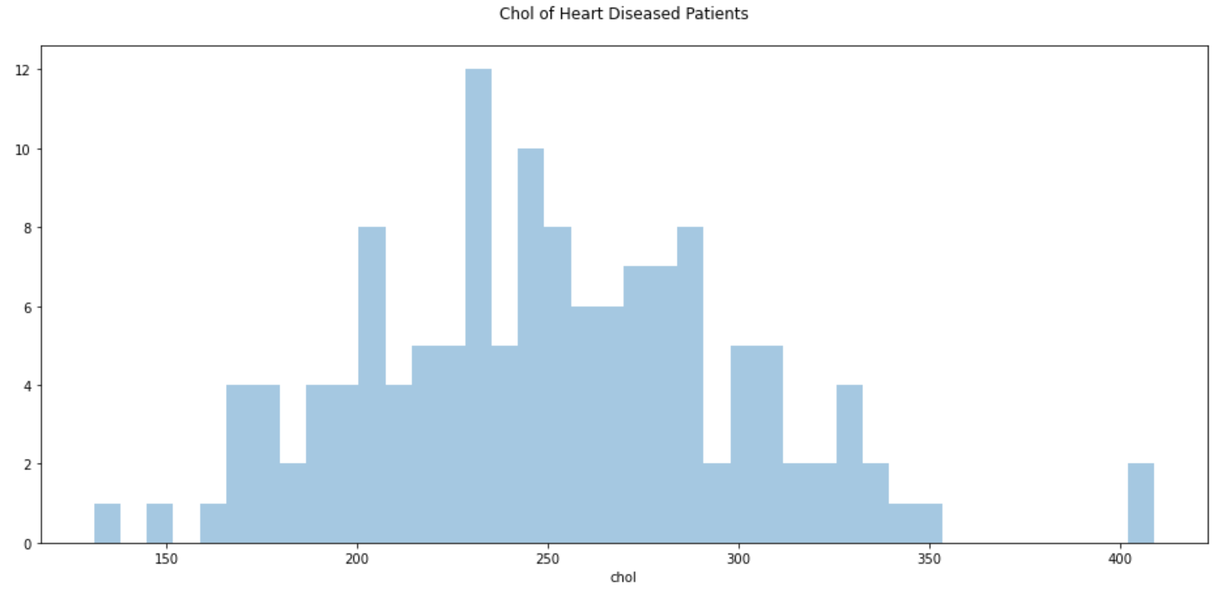
Out[19]:
```
Text(0.5, 1.0, 'Age of Heart Diseased Patients\n')
```



Heart Disease is very common in the seniors which is composed of age group 60 and above and common among adults which belong to the age group of 41 to 60. But it's rare among the age group of 19 to 40 and very rare among the age group of 0 to 18.

In [20]:
```
plt.figure(figsize=(16,7))
sns.distplot(data[data['target']==0]['chol'],kde=False,bins=40)
plt.title('Chol of Heart Diseased Patients\n')
```

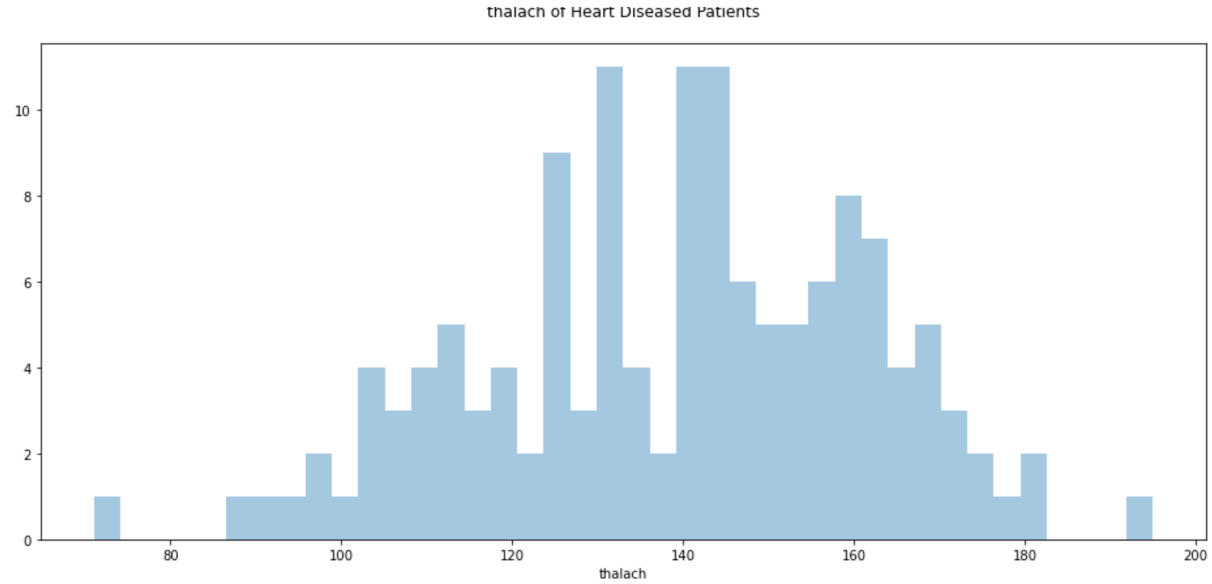
Out[20]:
```
Text(0.5, 1.0, 'Chol of Heart Diseased Patients\n')
```



Total cholesterol LDL — ‘bad cholesterol” HDL — ‘good cholesterol” In adults, the total cholesterol levels are considered desirable less than 200 milligram per decilitre (mg / dL). Borderlines are considered to be high between 200 to 239 mg / dL and 240 mg / dL and above. LDL should contain less than 100 mg / dL of cholesterol. 100 mg / dl rates for individuals without any health issue are appropriate but may be more relevant for those with cardiac problems or risk factors for heart disease. The levels are borderline moderate between 130 and 159 mg / dL and moderate between 160 and 189 mg / dL. The reading is very high at or above 190 mg / dL. Levels of HDL are to be maintained higher. The risk factor for cardiovascular diseases is called a reading less than 40 mg / dL. Borderline low is considered to be between 41 mg / dL and 59 mg / dL. The HDL level can be measured with a maximum of 60 mg / dL.

In [21]:
```
plt.figure(figsize=(16,7))
sns.distplot(data[data['target']==0]['thalach'],kde=False,bins=40)
plt.title('thalach of Heart Diseased Patients\n')
```

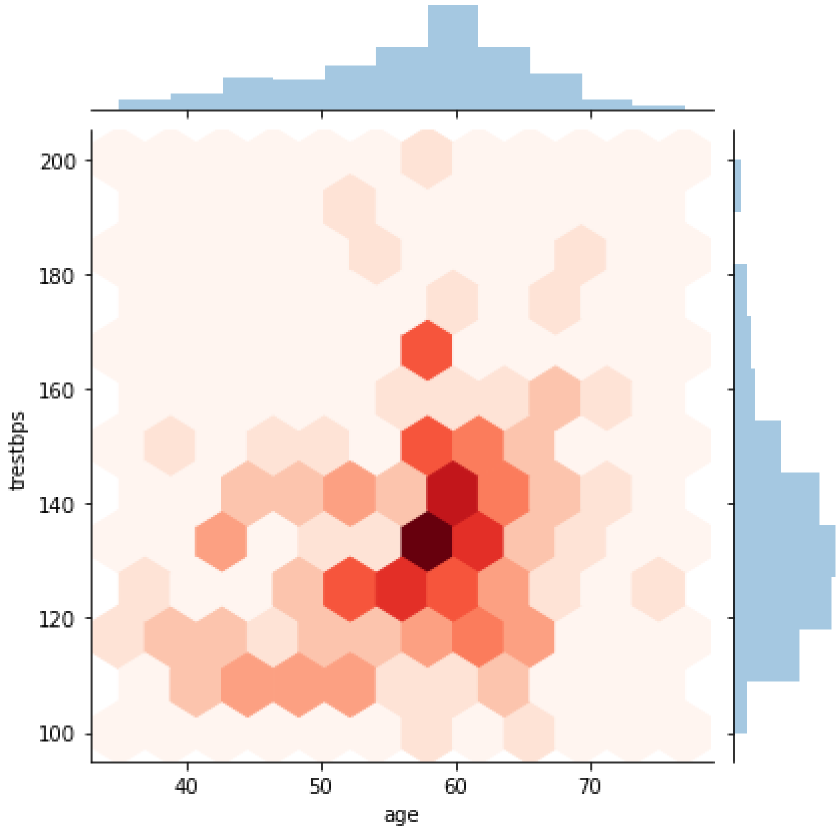
Out[21]:
```
Text(0.5, 1.0, 'thalach of Heart Diseased Patients\n')
```



In [22]:
```
df3 = data[data['target'] == 0 ][['age', 'sex', 'cp', 'trestbps', 'cho
l', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']]
#target 0 - people with heart disease
pal = sns.light_palette("blue", as_cmap=True)
print('Age vs trestbps(Heart Diseased Patinets)')
sns.jointplot(data=df3,
              x='age',
              y='trestbps',
              kind='hex',
              cmap='Reds'

              )
```

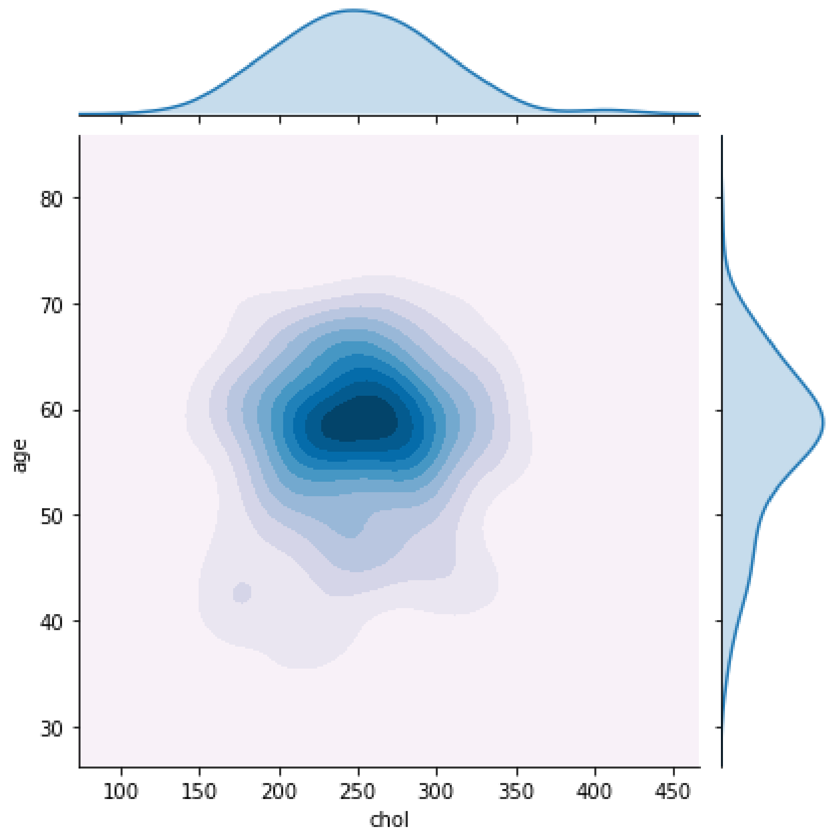
```
Age vs trestbps(Heart Diseased Patinets)
```

Out[22]:
```
<seaborn.axisgrid.JointGrid at 0x27ffe4fcd0>
```



In [23]:
```
sns.jointplot(data=df3,
              x='chol',
              y='age',
              kind='kde',
              cmap='PuBu'
              )
```

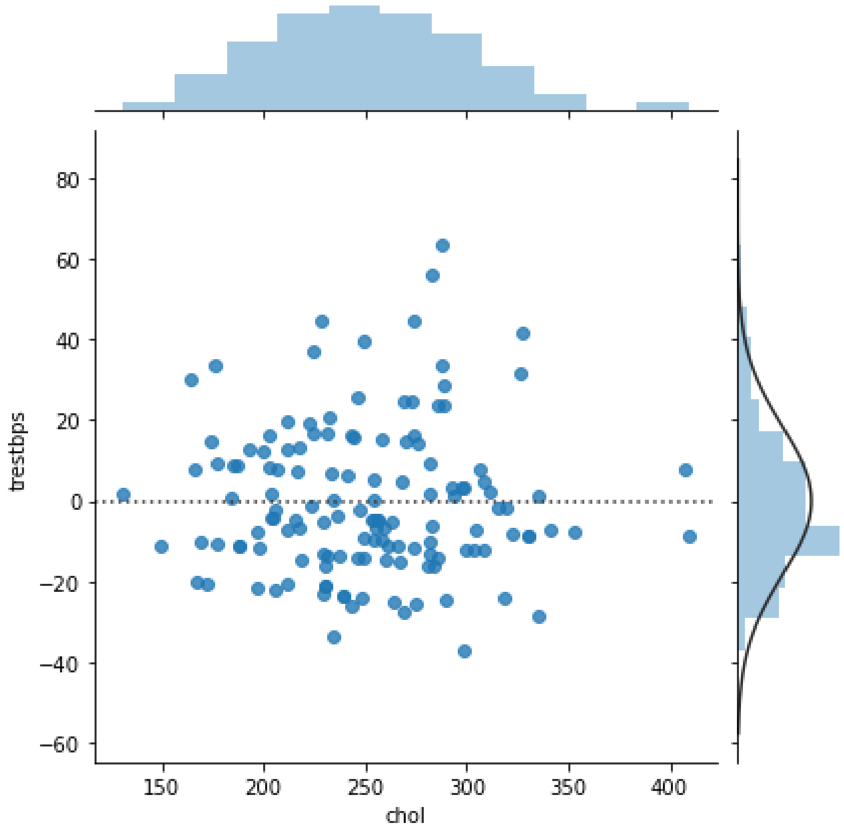
Out[23]: <seaborn.axisgrid.JointGrid at 0x27817b55e0>



Joint plots in seaborn helps us to understand the trend seen among two features. As observed from the above plot we can see that most of the Heart diseased patients in their age of upper 50s or lower 60s tend to have Cholesterol between 200mg/dl to 300mg/dl.

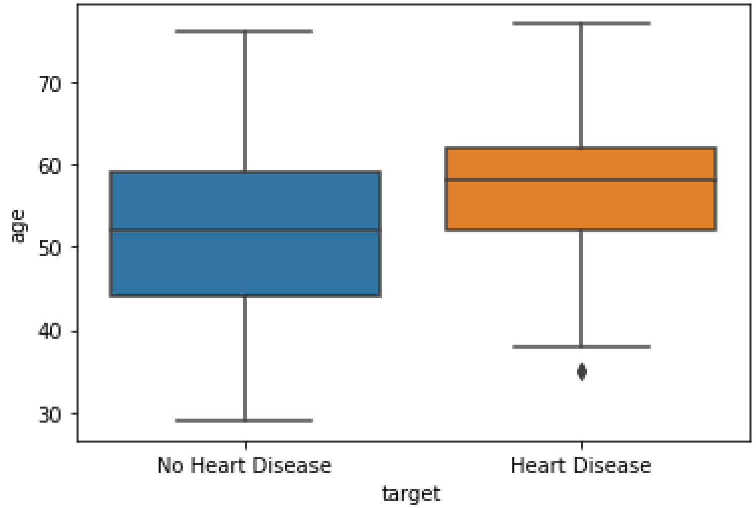
In [24]:
```
sns.jointplot(data=df3,
              x='chol',
              y='trestbps',
              kind='resid',

              )
```

Out[24]: <seaborn.axisgrid.JointGrid at 0x27ffeb5160>



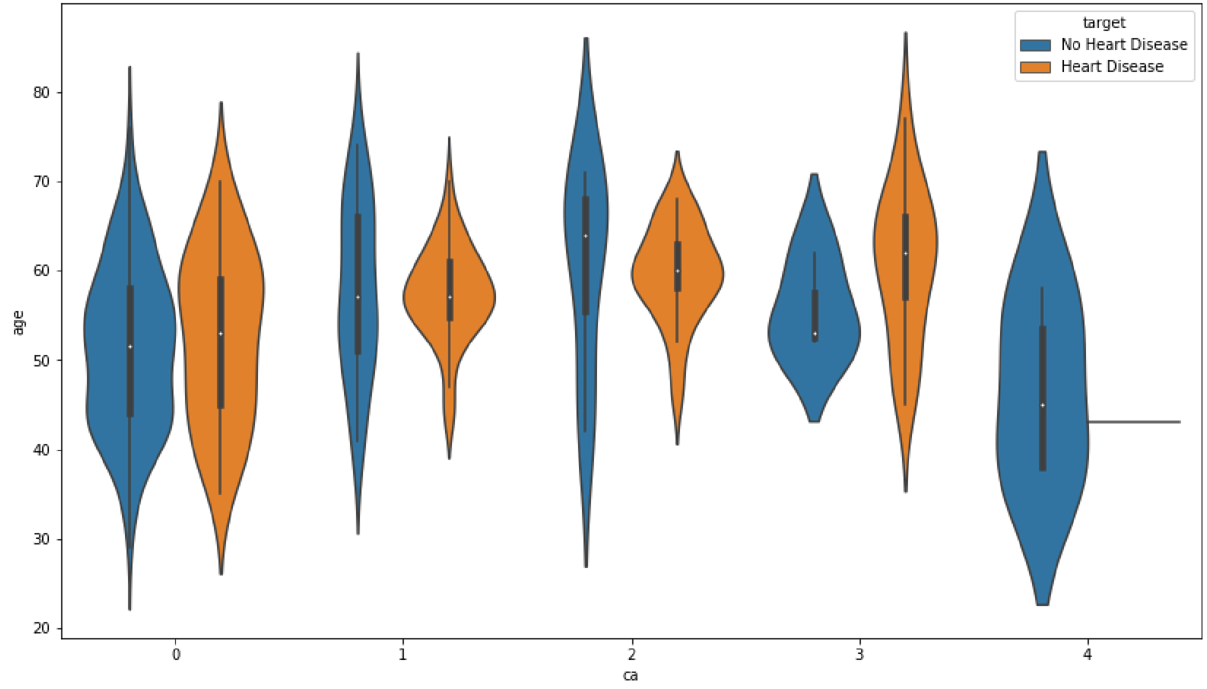
In [25]:
```
sns.boxplot(data=df2,x='target',y='age')
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x27811fd850>



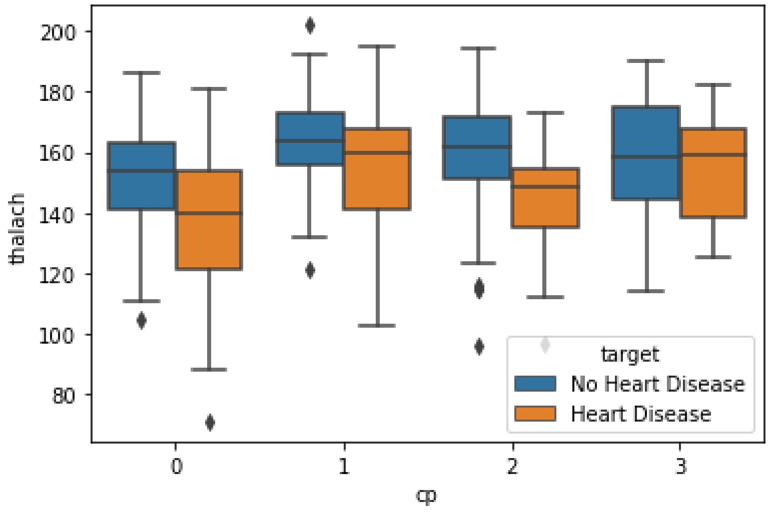
In [26]:
```
plt.figure(figsize=(14,8))
sns.violinplot(data=df2,x='ca',y='age',hue='target')
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x27822d0af0>



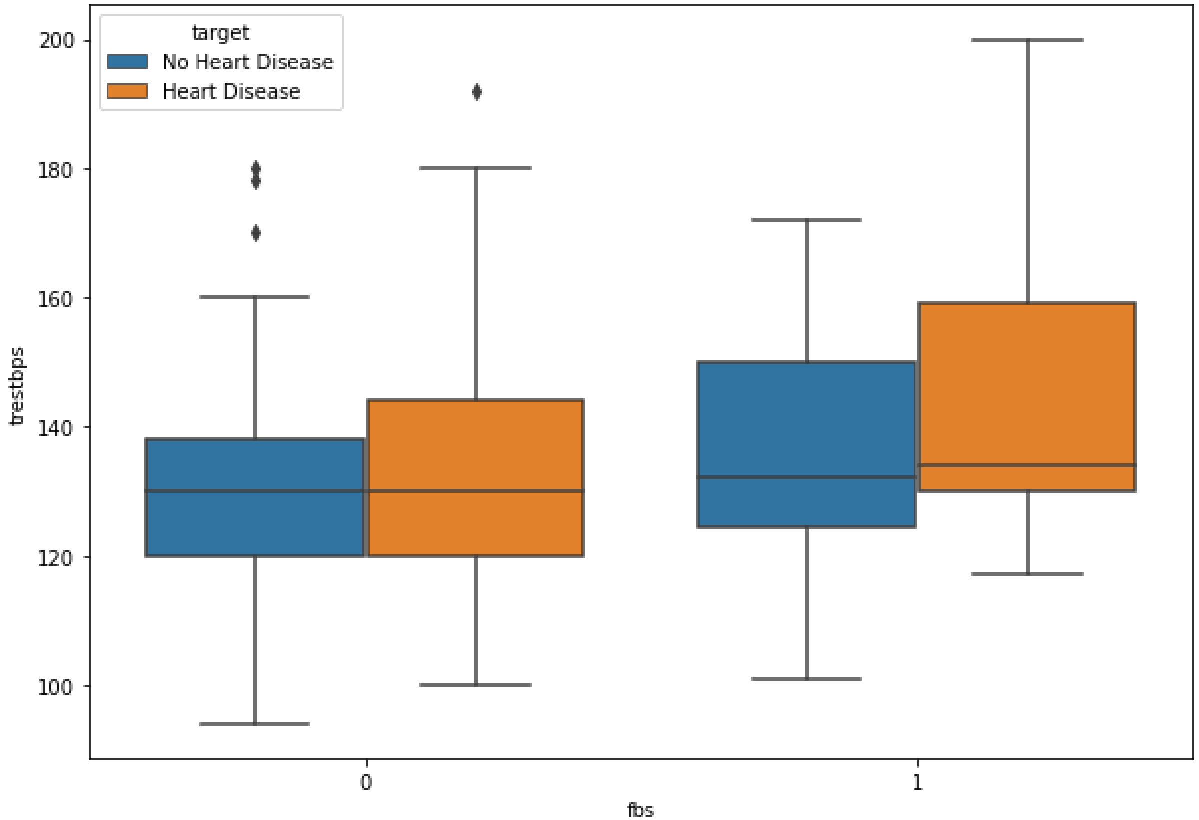
In [27]:
```
sns.boxplot(data=df2,x='cp',y='thalach',hue='target')
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x278322e490>

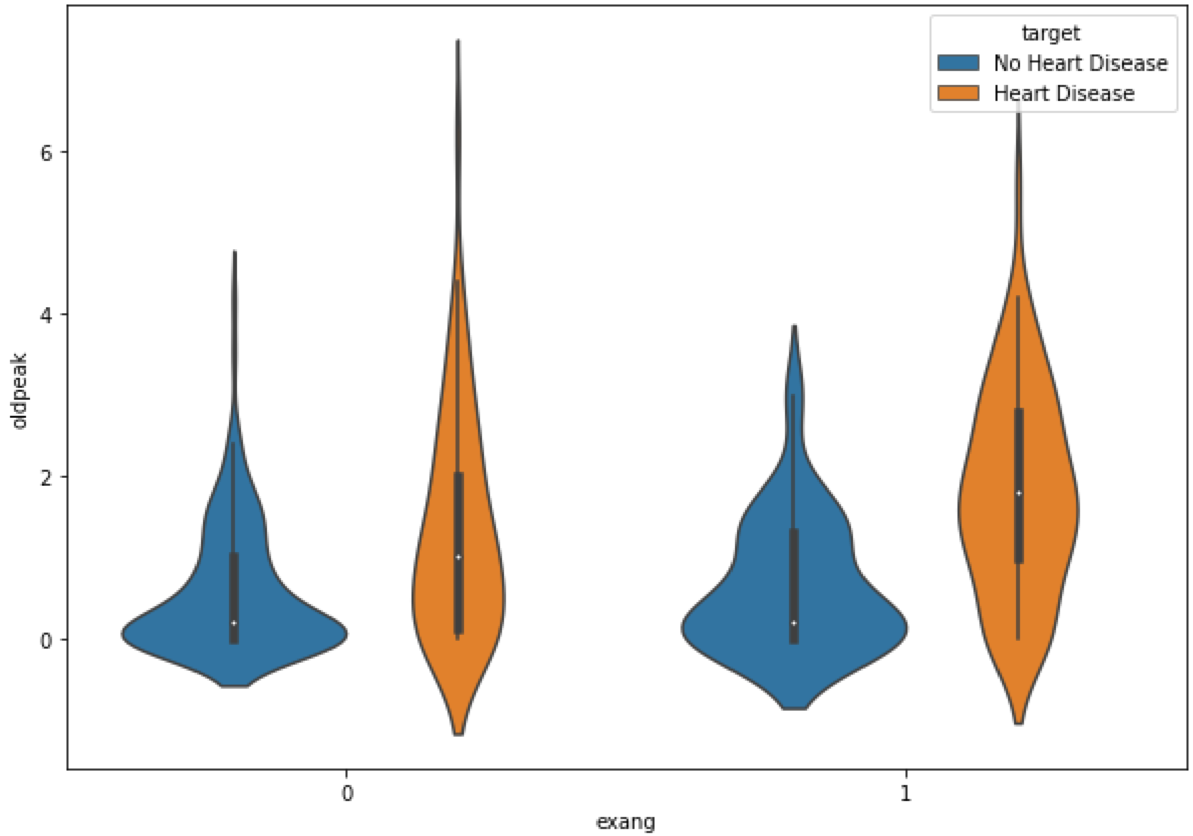


In [28]:
```
plt.figure(figsize=(10,7))
sns.boxplot(data=df2,x='fbs',y='trestbps',hue='target')
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x278510f7c0>

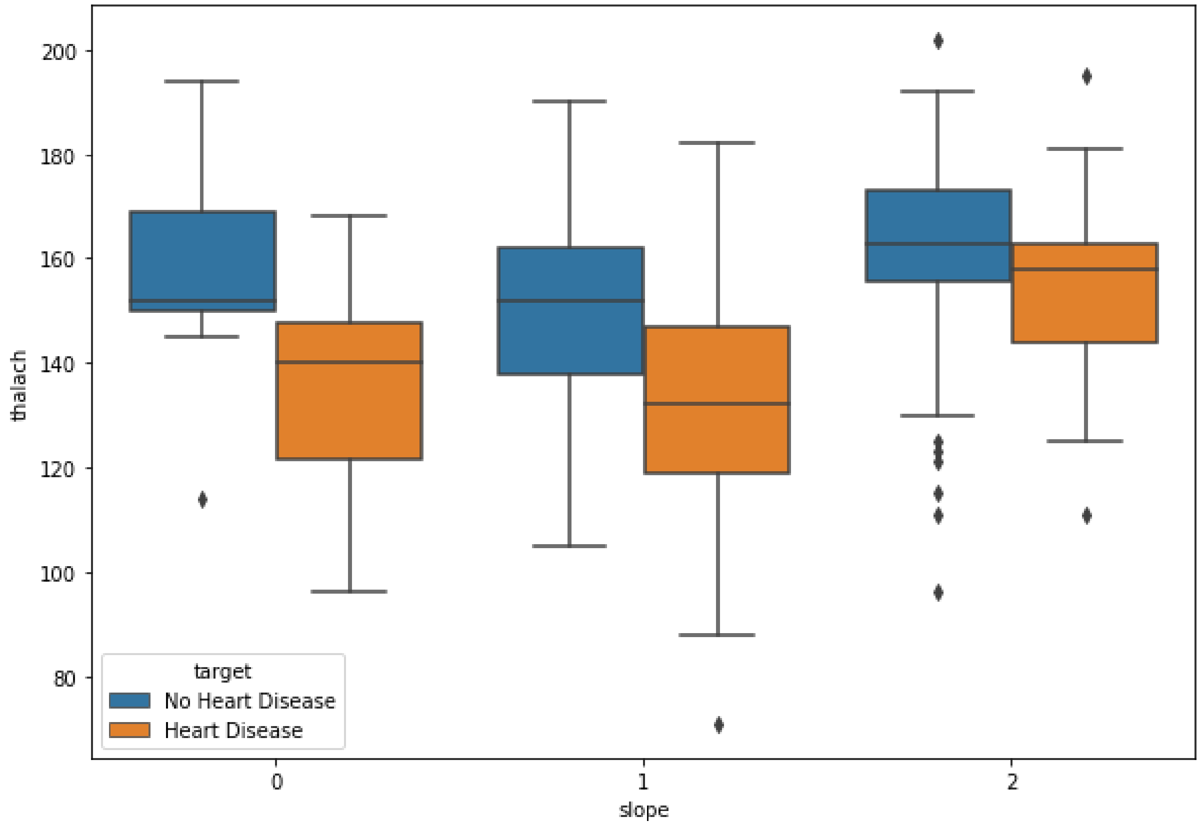



```
In [29]: plt.figure(figsize=(10,7))
         sns.violinplot(data=df2,x='exang',y='oldpeak',hue='target')

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x2785134fd0>
```



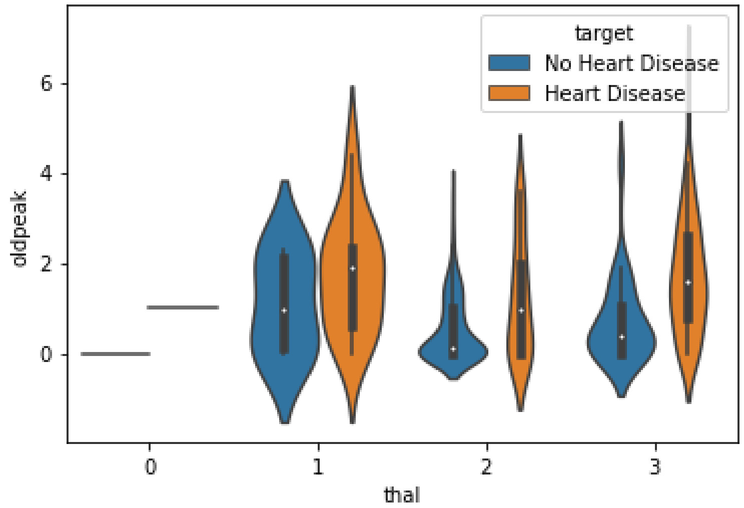
In [30]:
```
plt.figure(figsize=(10,7))
sns.boxplot(data=df2,x='slope',y='thalach',hue='target')
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x27851bf220>



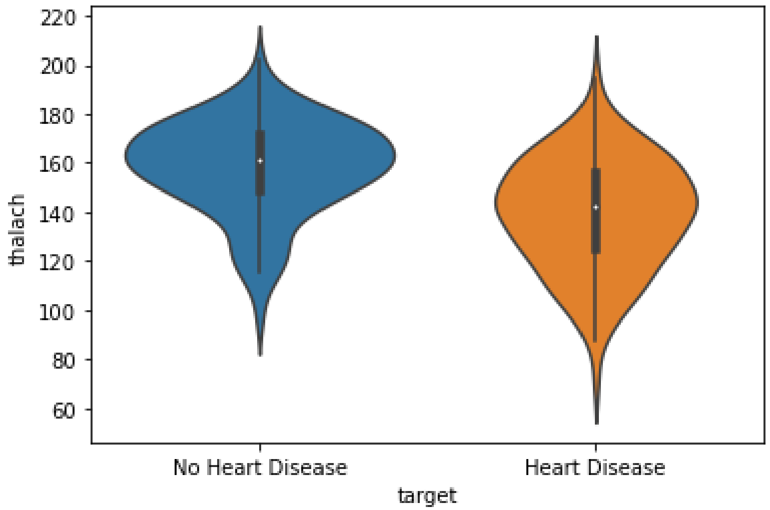
In [31]:
```
sns.violinplot(data=df2,x='thal',y='oldpeak',hue='target')
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x278549cd60>



In [32]:
```
sns.violinplot(data=df2,x='target',y='thalach')
```

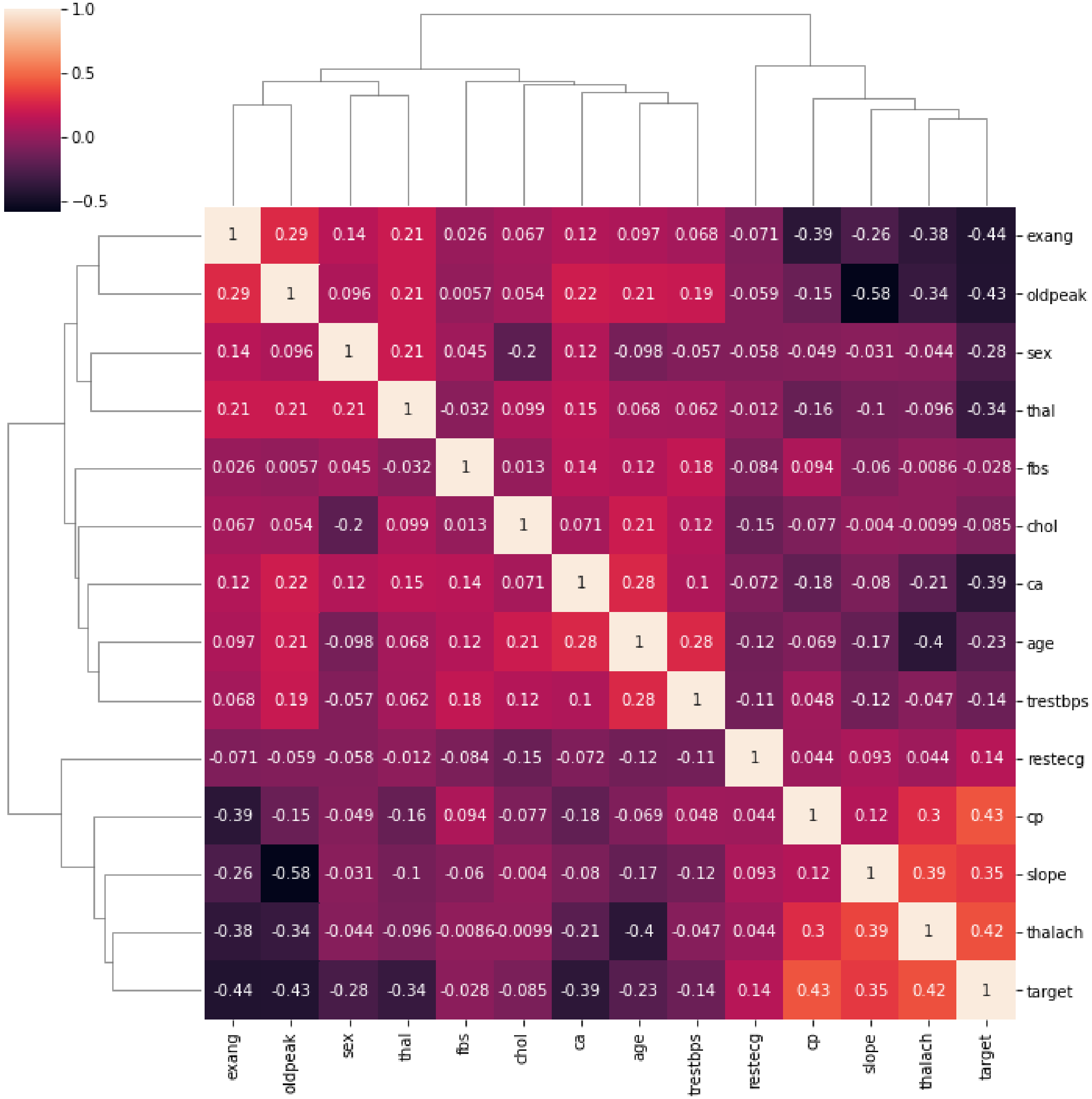
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x2785527f70>



Clusterplot

```
In [33]: sns.clustermap(data.corr(),annot=True)
```

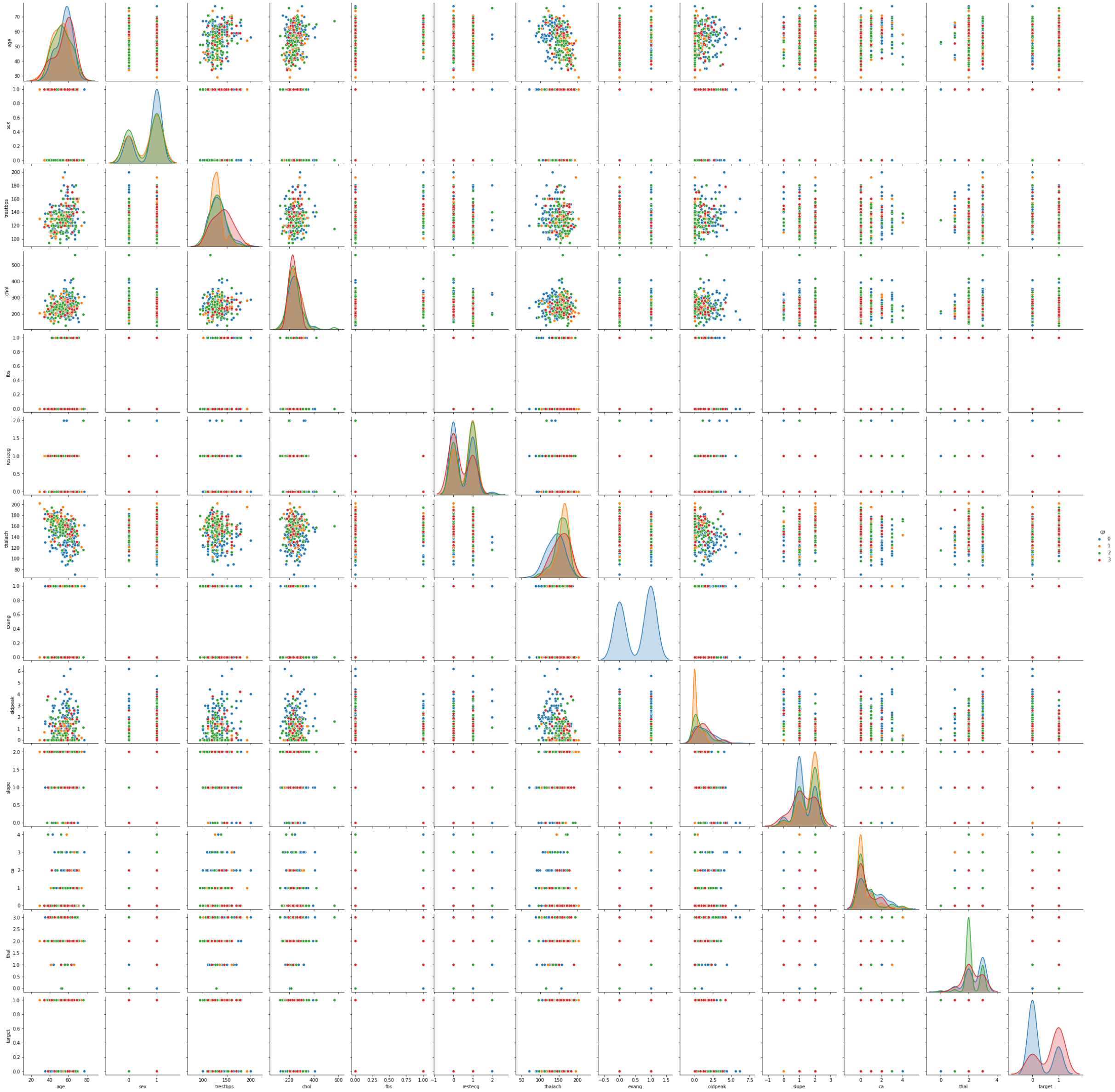
```
Out[33]: <seaborn.matrix.ClusterGrid at 0x27822c7f40>
```



Pairplot

In [34]:
```
sns.pairplot(data,hue='cp')
```

Out[34]:
```
<seaborn.axisgrid.PairGrid at 0x2785643cd0>
```



Train Test Split

In [37]:
```
from sklearn.model_selection import train_test_split
```

In [38]:
```
predictors = data.drop("target",axis=1)
target = data["target"]
X_train,X_test,Y_train,Y_test = train_test_split(predictors,target,test
_size=0.20,random_state=0)
```

In [39]:
```
X_train.shape
```

Out[39]:
```
(242, 13)
```

In [40]:
```
X_test.shape
```

Out[40]:
```
(61, 13)
```

In [41]:
```
Y_train.shape
```

Out[41]:
```
(242,)
```

In [42]:
```
Y_test.shape
```

Out[42]:
```
(61,)
```

In [60]:
```
from sklearn.metrics import accuracy_score
import statsmodels.api as sn
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import model_selection
from sklearn.metrics import classification_report,roc_auc_score,roc_cur
ve
from sklearn.metrics import classification_report
import pickle
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_fac
tor
from statsmodels.tools.tools import add_constant
```

Model Fitting

Naive Bayes

In [45]:
```
nb = GaussianNB()

Y_train=Y_train.astype('int')

nb.fit(X_train,Y_train)


Y_pred_nb = nb.predict(X_test)
```

In [46]:
```
Y_pred_nb.shape
```

Out[46]:
```
(61,)
```

In [47]:
```
# build confusion metrics
CM=pd.crosstab(Y_test,Y_pred_nb)
CM
```

Out[47]:

col_0	0	1
target		
0	21	6
1	3	31

In [48]:
```
#let us save TP, TN, FP, FN
TN=CM.iloc[0,0]
FP=CM.iloc[0,1]
FN=CM.iloc[1,0]
TP=CM.iloc[1,1]
```

In [49]:
```
#check accuracy of model
```

```
score_nb=((TP+TN)*100)/(TP+TN+FP+FN)
score_nb
```

Out[49]: 85.24590163934427

In [50]:
```
# check false negative rate of the model
fnr=FN*100/(FN+TP)
fnr
```

Out[50]: 8.823529411764707

K - Nearest Neighbour

for neighbors = 7

In [62]:
```
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,Y_train)
Y_pred_knn=knn.predict(X_test)
```

In [63]:
```
Y_pred_knn.shape
```

Out[63]: (61,)

In [64]:
```
score_knn = round(accuracy_score(Y_pred_knn,Y_test)*100,2)

print("The accuracy score achieved using KNN is: "+str(score_knn)+" %")
```

```
The accuracy score achieved using KNN is: 67.21 %
```

for neighbors = 4

In [70]:
```
knn_model=KNeighborsClassifier(n_neighbors=4).fit(X_train,Y_train)
knn_predictions=knn_model.predict(X_test)
```

In [71]:
```
# build confusion metrics
CM=pd.crosstab(Y_test,knn_predictions)
CM
```

Out[71]:

col_0	0	1
target		
0	19	8
1	14	20

In [74]:
```
k_range = range(1, 26)

# We can create Python dictionary using [] or dict()
scores = []
from sklearn import metrics
# We use a loop through the range 1 to 26
# We append the scores in the dictionary
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, Y_train)
    y_pred = knn.predict(X_test)
    scores.append(metrics.accuracy_score(Y_test, y_pred))

print(scores)
```

```
[0.5245901639344263, 0.5901639344262295, 0.639344262295082, 0.639344262
295082, 0.639344262295082, 0.6557377049180327, 0.6721311475409836, 0.68
85245901639344, 0.6721311475409836, 0.6557377049180327, 0.7049180327868
853, 0.6721311475409836, 0.7213114754098361, 0.6721311475409836, 0.6721
311475409836, 0.6721311475409836, 0.7213114754098361, 0.688524590163934
4, 0.7049180327868853, 0.6885245901639344, 0.7049180327868853, 0.688524
5901639344, 0.6885245901639344, 0.6557377049180327, 0.6885245901639344]
```

In [75]:
```
#let us save TP, TN, FP, FN
TN=CM.iloc[0,0]
FP=CM.iloc[0,1]
FN=CM.iloc[1,0]
TP=CM.iloc[1,1]
```

In [81]:
```
#check accuracy of model
score_knn_4=((TP+TN)*100)/(TP+TN+FP+FN)
```

```
score_knn_4
```

Out[81]: 63.9344262295082

In [79]:
```
# check false negative rate of the model
fnr=FN*100/(FN+TP)
fnr
```

Out[79]: 41.1764705882353

Logistic Regression

In [84]:
```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()

lr.fit(X_train,Y_train)

Y_pred_lr = lr.predict(X_test)
```

In [85]:
```
Y_pred_lr.shape
```

Out[85]: (61,)

In [86]:
```
score_lr = round(accuracy_score(Y_pred_lr,Y_test)*100,2)

print("The accuracy score achieved using Logistic Regression is: "+str(
score_lr)+" %")
```

```
The accuracy score achieved using Logistic Regression is: 85.25 %
```

Final Score

In [88]:
```
scores = [score_lr,score_nb,score_knn]
```

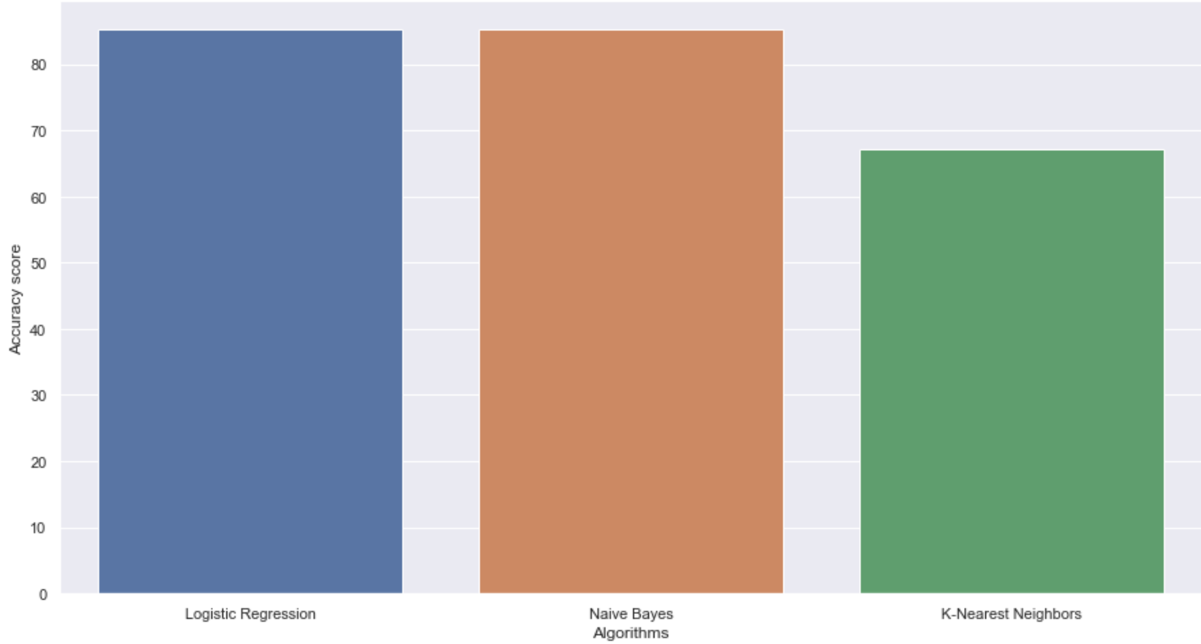
```
algorithms = ["Logistic Regression","Naive Bayes","K-Nearest Neighbors"
]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+st
r(scores[i])+" %")
```

```
The accuracy score achieved using Logistic Regression is: 85.25 %
The accuracy score achieved using Naive Bayes is: 85.24590163934427 %
The accuracy score achieved using K-Nearest Neighbors is: 67.21 %
```

In [89]:

```
sns.set(rc={'figure.figsize':(15,8)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

Out[89]: <matplotlib.axes._subplots.AxesSubplot at 0x279108e2b0>



We observe that, we can achieve the best accuracy of 85.25% using Logistic Regression

In [ ]:

```

```