Name-Anjali.N

Project name-Prediction of Heart disease detection

In [ ]:
```
#
```

Import Required Libraries

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

Import dataset

In [ ]:
```
dataset = pd.read_csv("/content/drive/MyDrive/TCR Internship Project/heart.csv")
```

In [ ]:
```
dataset
```

Out[ ]:

	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

303 rows × 14 columns

Shape of dataset

In [ ]:
```
dataset.shape
```

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

Some Operations on dataset

In [ ]:
```
dataset.head()
```

Out[ ]:

	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 [ ]:
```
dataset.tail()
```

Out[ ]:

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

In [ ]:

```
type(dataset)
```

Out[ ]:

```
pandas.core.frame.DataFrame
```

In [ ]:

```
dataset.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 [ ]:

```
dataset.describe()
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
count	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
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	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
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	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
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000

In [ ]:

```
dataset.columns
```

Out[ ]:

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

Checking total number of NA values

In [ ]:

```
dataset.isna().sum()
```

Out[ ]:

```
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
```

Checking total number of NULL values

In [ ]:
```
dataset.isnull().sum()
```

Out[ ]:
```
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 [ ]:
```
#
```

Exploratory Data Analysis (EDA)

Analysing the 'target' variable

In [ ]:
```
dataset.target.describe()
```

Out[ ]:
```
count    303.000000
mean       0.544554
std        0.498835
min        0.000000
25%        0.000000
50%        1.000000
75%        1.000000
max        1.000000
Name: target, dtype: float64
```

In [ ]:
```
dataset.target.unique()
```

Out[ ]:
```
array([1, 0])
```

In [ ]:
```
#Checking correlation between columns
dataset.corr()["target"].abs().sort_values(ascending=False)
```

Out[ ]:
```
target      1.000000
exang       0.436757
cp          0.433798
oldpeak     0.430696
thalach     0.421741
ca          0.391724
slope       0.345877
thal        0.344029
sex         0.280937
age         0.225439
trestbps    0.144931
restecg     0.137230
chol        0.085239
fbs         0.028046
Name: target, dtype: float64
```

In [ ]:

```
#This shows that most columns are moderately correlated with target, but 'fbs' is very weakly correlated.
```

In [ ]:

```
dataset.target.value_counts()
```

Out[ ]:

```
1    165
0    138
Name: target, dtype: int64
```

Patient without heart problems - labeled as 0

Patient with heart problems - labeled as 1

In [ ]:

```
print("Percentage of patients without heart problems: "+str(round(138*100/303,2)))
print("Percentage of patients with heart problems: "+str(round(165*100/303,2)))
```

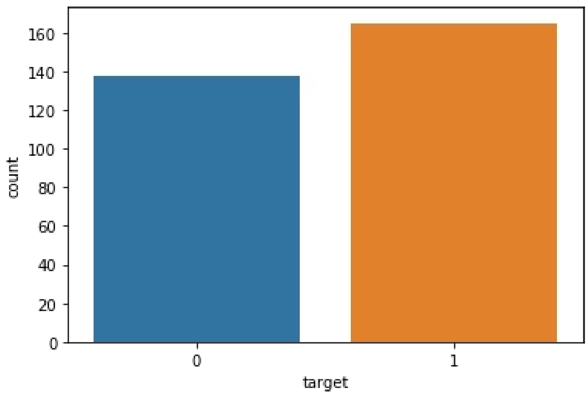
```
Percentage of patient without heart problems: 45.54
Percentage of patient with heart problems: 54.46
```

In [ ]:

```
y = dataset["target"]
sns.countplot(y)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89a51e5d90>
```

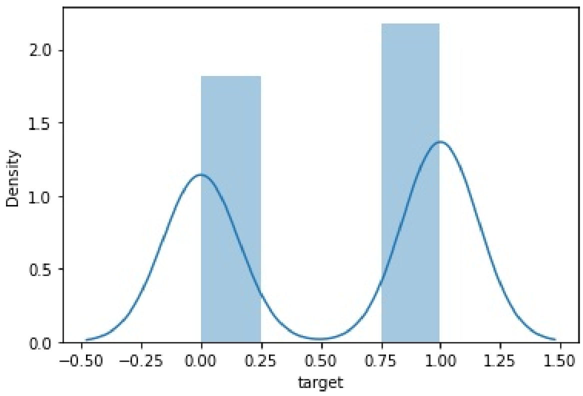


In [ ]:

```
sns.distplot(dataset['target'])
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8991c58910>
```



Analysing the 'sex' variable

In [ ]:

```
dataset.sex.value_counts()
```

Out[ ]:

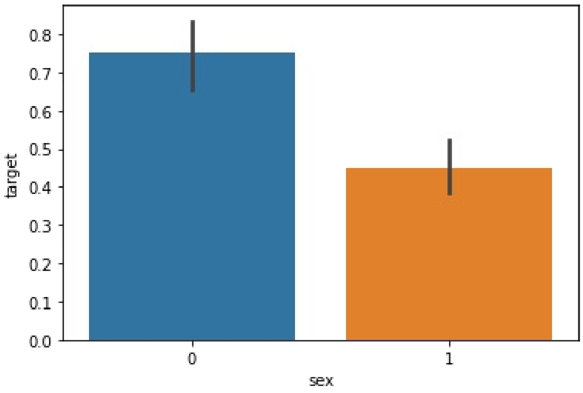
```
1    207
0     96
Name: sex, dtype: int64
```

In [ ]:

```
sns.barplot(dataset["sex"],y)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89a5127550>
```



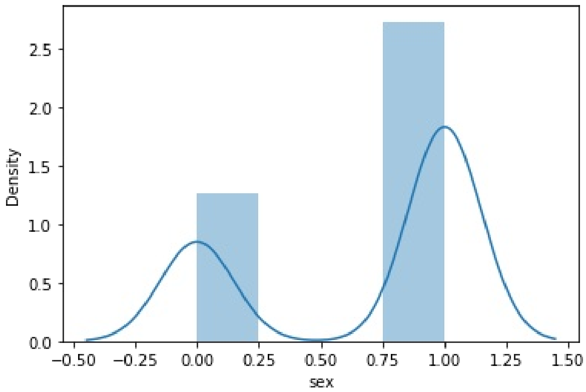
We notice that the 'sex' feature has 2 unique features.

In [ ]:

```
sns.distplot(dataset['sex'])
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8991cd29d0>
```



Analysing the 'cp' variable

In [ ]:

```
dataset.cp.value_counts()
```

Out[ ]:

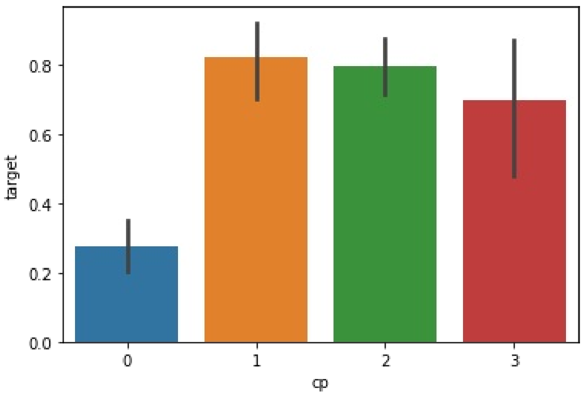
```
0    143
2     87
1     50
3     23
Name: cp, dtype: int64
```

In [ ]:

```
sns.barplot(dataset["cp"],y)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89a4c5a810>
```



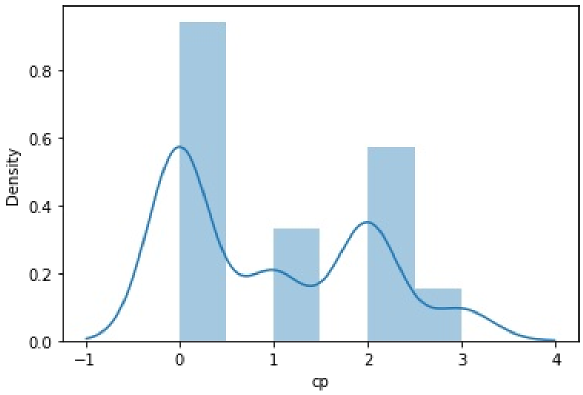
The CP feature has values from 0 to 3.We notice, that chest pain of '0', are much less likely to have heart problems

In [ ]:

```
sns.distplot(dataset['cp'])
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8991d89c90>
```



Analysing the 'age' variable

In [ ]:

```
dataset.age.value_counts()
```

Out[ ]:

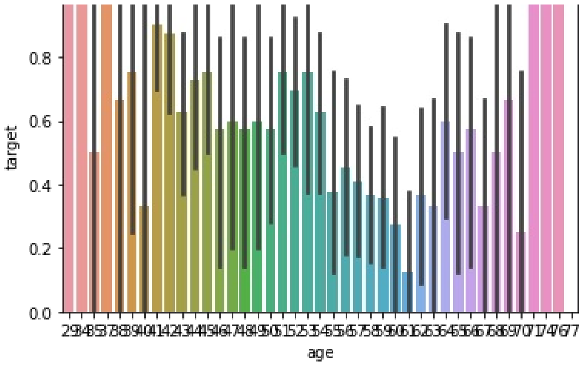
```
58    19
57    17
54    16
59    14
52    13
51    12
62    11
44    11
60    11
56    11
64    10
41    10
63     9
67     9
55     8
45     8
42     8
53     8
61     8
65     8
43     8
66     7
50     7
48     7
46     7
49     5
47     5
39     4
35     4
68     4
70     4
40     3
71     3
69     3
38     3
34     2
37     2
77     1
76     1
74     1
29     1
Name: age, dtype: int64
```

In [ ]:

```
sns.barplot(dataset["age"],y)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89a4bd5a90>
```



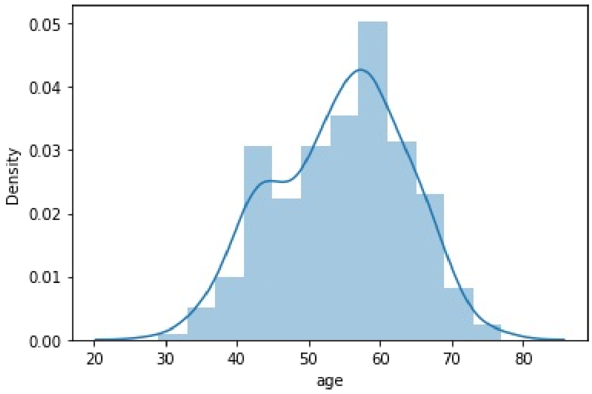
Nothing special here.

In [ ]:

```
sns.distplot(dataset['age'])
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8991d66b90>
```



Analysing the 'trestbps' variable

In [ ]:

```
dataset.trestbps.value_counts()
```

Out[ ]:

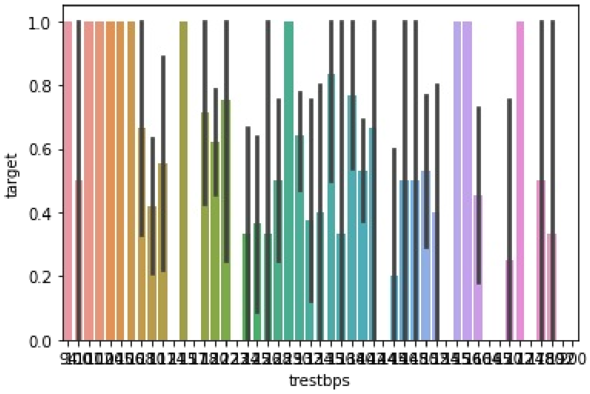
```
120    37
130    36
140    32
110    19
150    17
138    13
128    12
125    11
160    11
112     9
132     8
118     7
135     6
108     6
124     6
145     5
134     5
152     5
122     4
170     4
100     4
142     3
115     3
136     3
105     3
180     3
126     3
102     2
94      2
144     2
178     2
146     2
148     2
129     1
165     1
101     1
174     1
104     1
```

```
172     1
106     1
156     1
164     1
192     1
114     1
155     1
117     1
154     1
123     1
200     1
Name: trestbps, dtype: int64
```

In [ ]:

```python
sns.barplot(dataset["trestbps"],y)
```

Out[ ]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f89a4bbb850>`

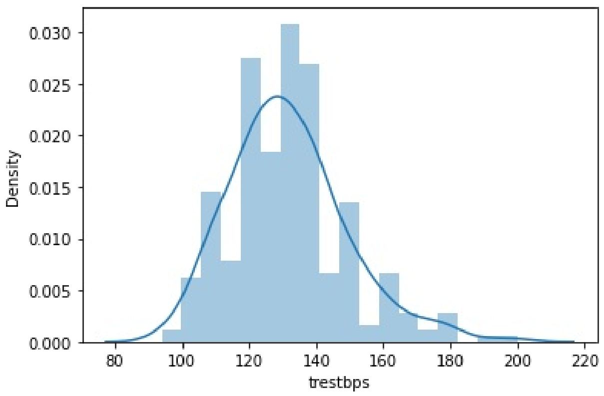


Nothing special here.

In [ ]:

```python
sns.distplot(dataset['trestbps'])
```

Out[ ]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f8991bd31d0>`



Analysing the 'chol' variable

In [ ]:

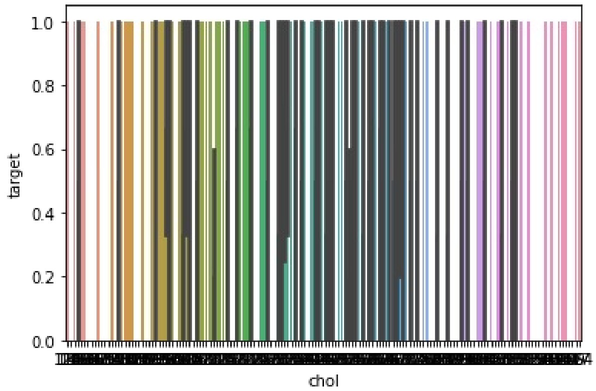
```python
dataset.chol.value_counts()
```

Out[ ]:

```
234    6
204    6
197    6
269    5
212    5
      ..
278    1
281    1
284    1
290    1
564    1
Name: chol, Length: 152, dtype: int64
```

In [ ]:
```
sns.barplot(dataset["chol"],y)
```

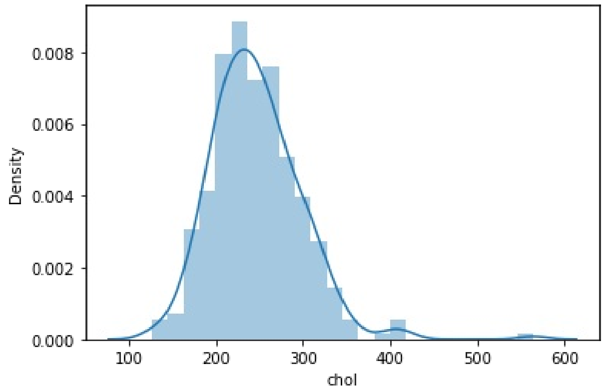
Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89a4736d10>
```



Nothing special here

In [ ]:
```
sns.distplot(dataset['chol'])
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8991b3ba10>
```



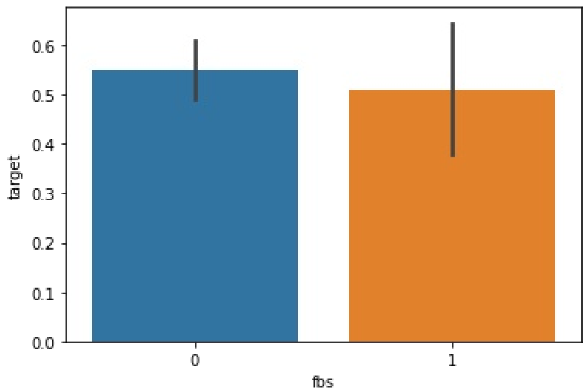
Analysing the 'fbs' variable

In [ ]:
```
dataset.fbs.value_counts()
```

Out[ ]:
```
0    258
1     45
Name: fbs, dtype: int64
```

In [ ]:
```
sns.barplot(dataset["fbs"],y)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89a4738490>
```

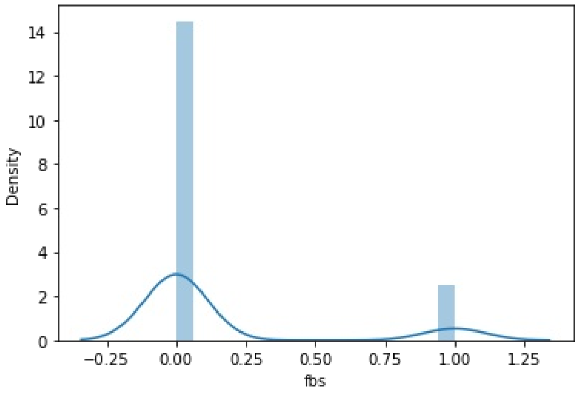


Not much difference here.

In [ ]:

```
sns.distplot(dataset['fbs'])
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8991aa1050>



Analysing the 'restecg' variable

In [ ]:

```
dataset.restecg.value_counts()
```

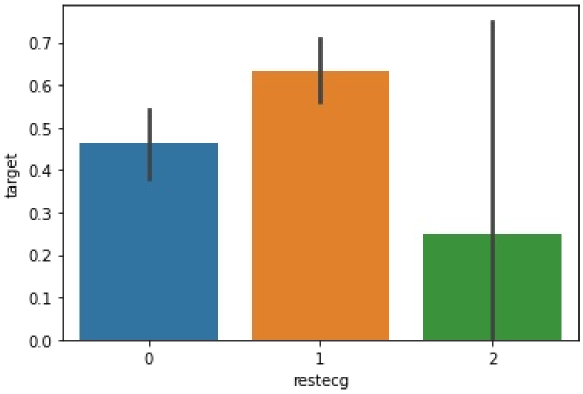
Out[ ]:

```
1    152
0    147
2      4
Name: restecg, dtype: int64
```

In [ ]:

```
sns.barplot(dataset["restecg"],y)
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f89a41a4f10>

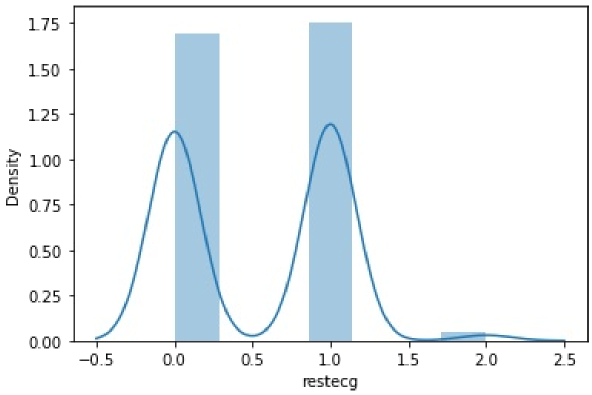


We realize that people with restecg '1' and '0' are much more likely to have a heart disease than with restecg '2'

In [ ]:

```
sns.distplot(dataset['restecg'])
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f89919bc990>



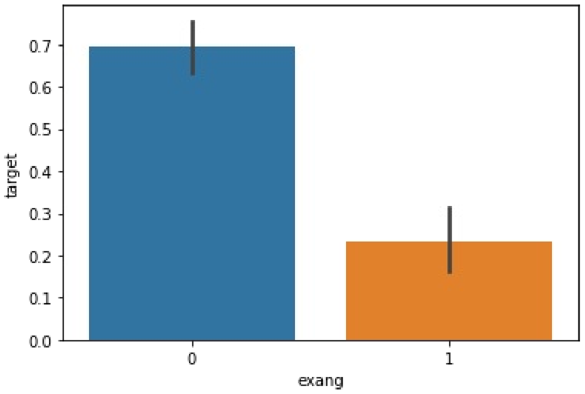
Analysing the 'exang' variable

In [ ]:
```
dataset.exang.value_counts()
```

Out[ ]:
```
0    204
1     99
Name: exang, dtype: int64
```

In [ ]:
```
sns.barplot(dataset["exang"],y)
```

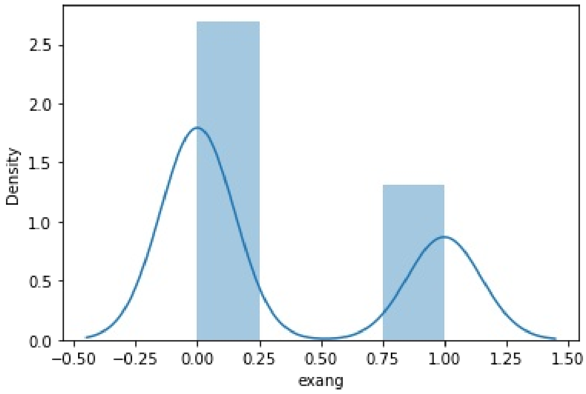
Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89a4120e90>
```



We notice here that people with exang=1, are much less likely to have heart problems.

In [ ]:
```
sns.distplot(dataset['exang'])
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8991934290>
```



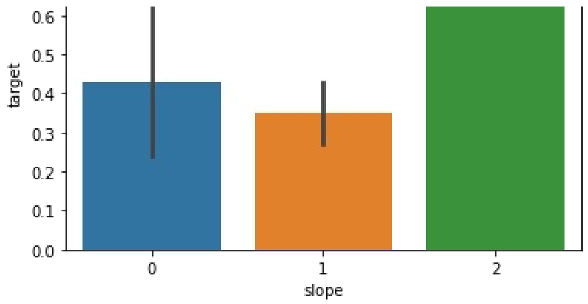
Analysing the 'slope' variable

In [ ]:
```
dataset.slope.value_counts()
```

Out[ ]:
```
2    142
1    140
0     21
Name: slope, dtype: int64
```

In [ ]:
```
sns.barplot(dataset["slope"],y)
```

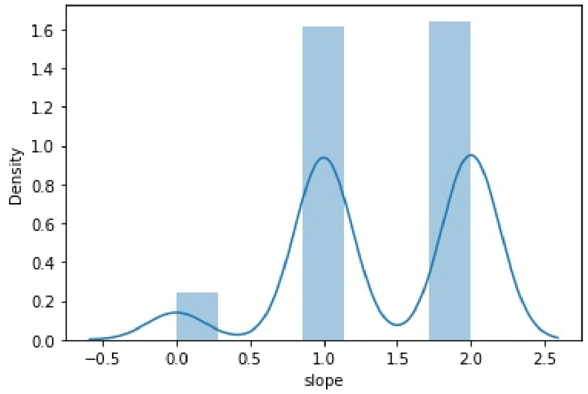
Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89a40e8b50>
```



We observe, that Slope '2' causes heart pain much more than Slope '0' and '1'

In [ ]:
```
sns.distplot(dataset['slope'])
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8991915b90>



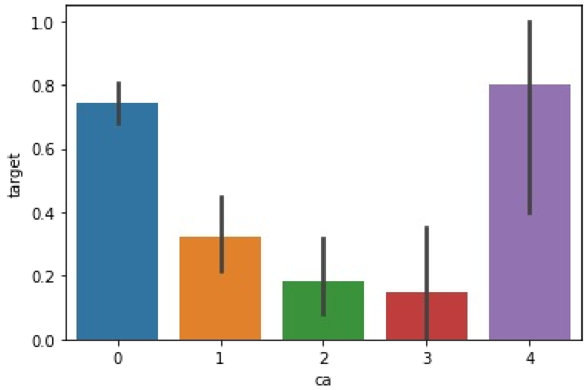
Analysing the 'ca' variable

In [ ]:
```
dataset.ca.value_counts()
```

Out[ ]:
```
0    175
1     65
2     38
3     20
4      5
Name: ca, dtype: int64
```

In [ ]:
```
sns.barplot(dataset["ca"],y)
```

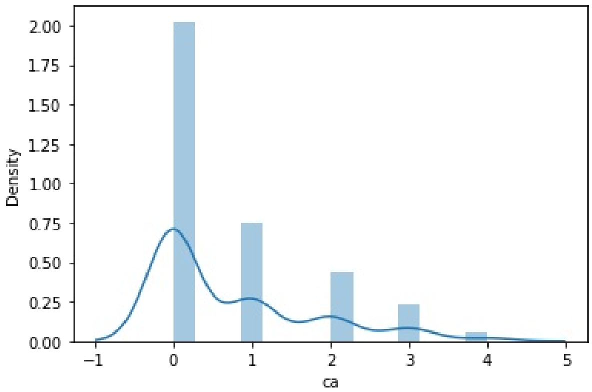
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f89a406fa90>



We notice that ca=4 has large number of heart patients.

In [ ]:
```
sns.distplot(dataset['ca'])
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f89918781d0>



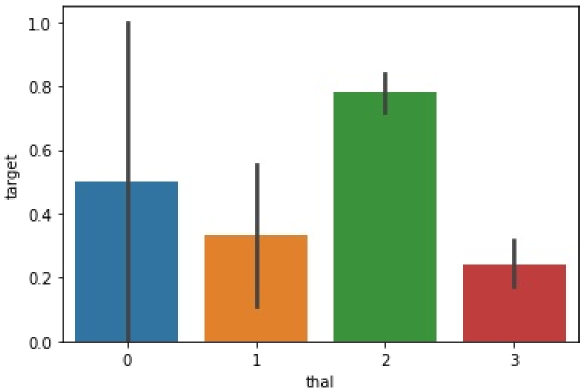
Analysing the 'thal' variable

In [ ]:
```
dataset.thal.value_counts()
```

Out[ ]:
```
2    166
3    117
1     18
0      2
Name: thal, dtype: int64
```

In [ ]:
```
sns.barplot(dataset["thal"],y)
```

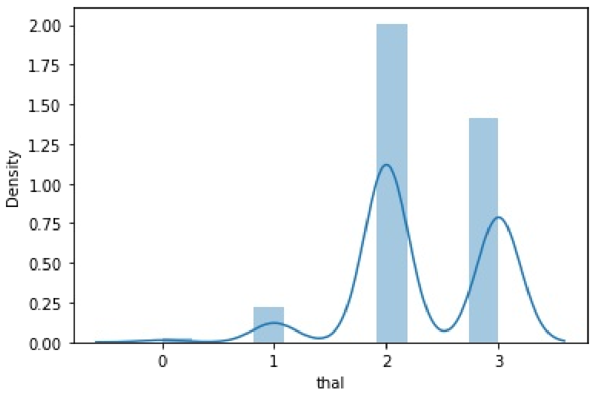
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f89a3fd8450>



thal=2 has large number of heart patients.

In [ ]:
```
sns.distplot(dataset['thal'])
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8991835f50>



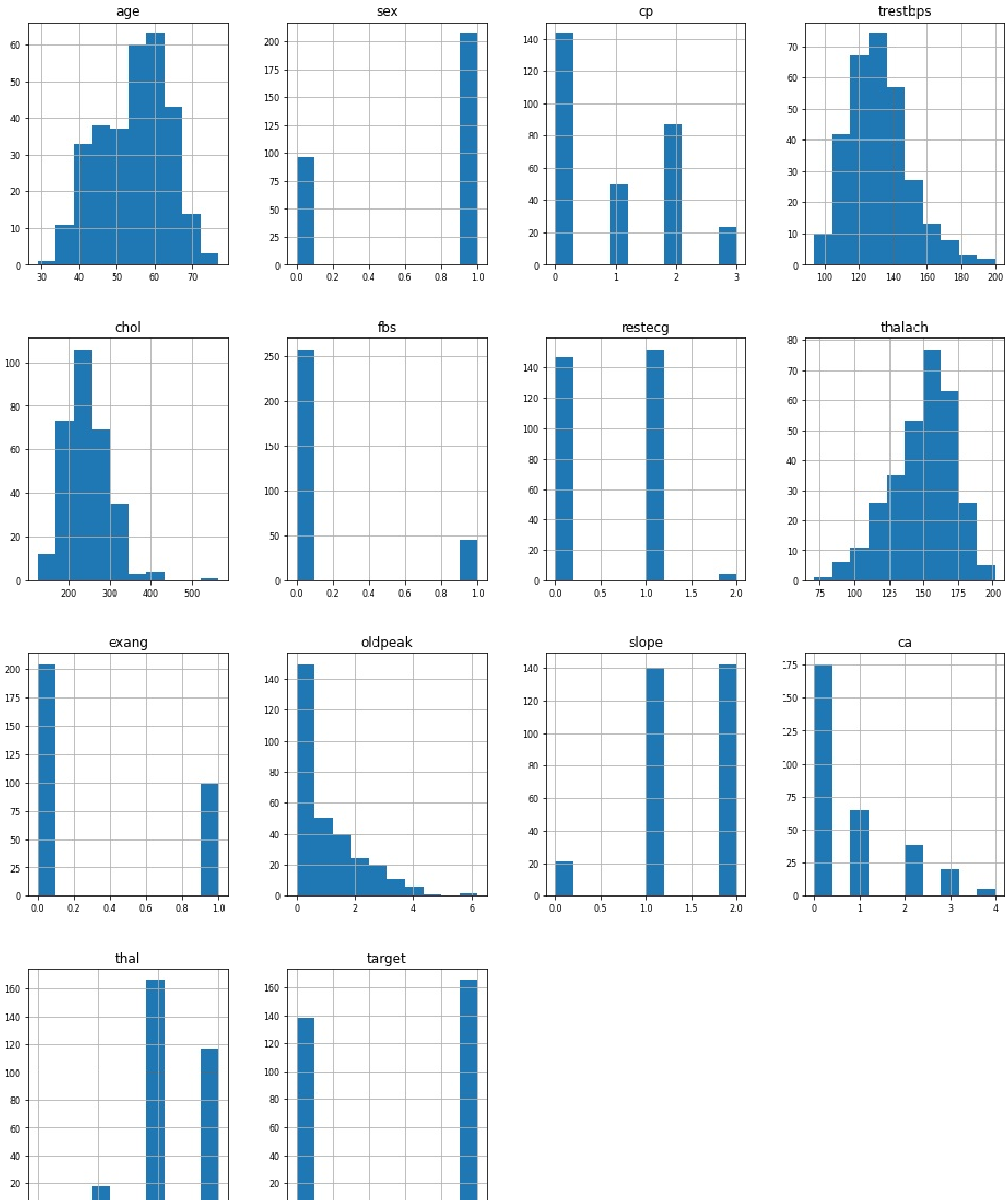
Get an overview distribution of each column

In [ ]:

```
dataset.hist(figsize=(16, 20), xlabelsize=8, ylabelsize=8)
```

Out[ ]:

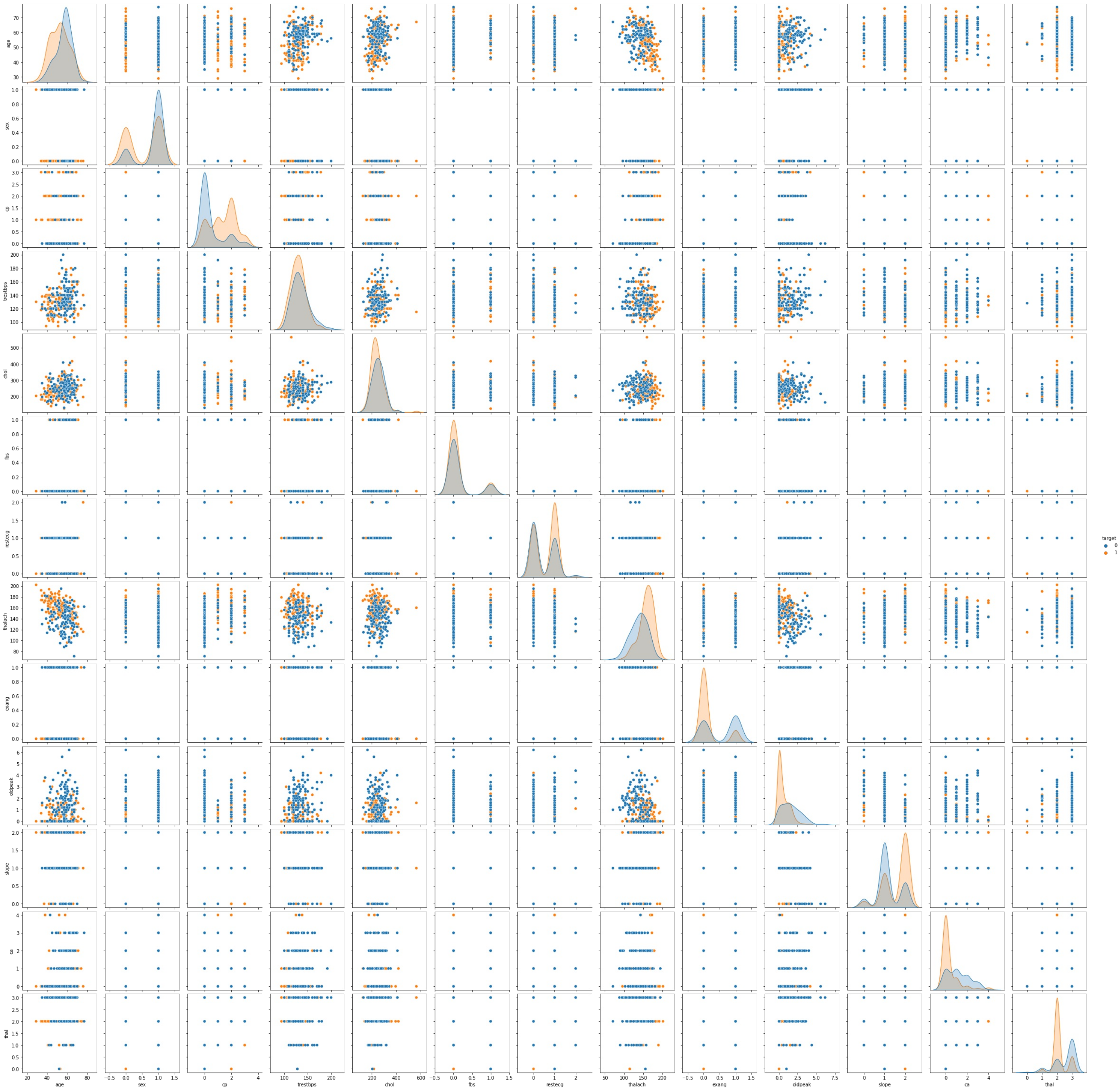
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3f11d10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3ecd390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3e83990>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3e39f90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3dfe2d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3db57d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3de9d50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3dab1d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3dab210>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3d63810>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3cdb150>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3c92650>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3c46b10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3bf2b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3bc0590>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f89a3b75a90>]],
      dtype=object)
```



In [ ]:

```
sns.pairplot(dataset, hue='target')
```

Out[ ]: <seaborn.axisgrid.PairGrid at 0x7f89a512fbd0>



Correlation heatmap

In [ ]:

```
dataset.corr()
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
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	0.068
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	0.210
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	-0.161
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	0.062
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	0.098
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	-0.032
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	-0.011
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	-0.096
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	0.206

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	0.210
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	-0.104
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	0.151
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	1.000
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	-0.344

In [ ]:

```
f, ax = plt.subplots(figsize=(15, 10))
sns.heatmap(dataset.corr(),annot=True,cmap='PiYG',linewidths=.5)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f89921d6f10>
```

Splitting the data - Train Test split

In [ ]:

```
from sklearn.model_selection import train_test_split
x = dataset.drop("target",axis=1)
y= dataset["target"]

X_train,X_test,Y_train,Y_test = train_test_split(x,y,test_size=0.20,random_state=0)
```

In [ ]:

```
X_train.shape
```

Out[ ]:

```
(242, 13)
```

In [ ]:

```
X_test.shape
```

Out[ ]:

```
(61, 13)
```

In [ ]:

```
Y_train.shape
```

Out[ ]:

```
(242,)
```

In [ ]:
```
Y_test.shape
```
Out[ ]:
```
(61,)
```

In [ ]:
```
from sklearn.metrics import accuracy_score
```

Logistic Regression

In [ ]:
```
from sklearn.linear_model import LogisticRegression
model_logistic_reg = LogisticRegression()
model_logistic_reg.fit(X_train,Y_train)
Y_pred_logistic_reg = model_logistic_reg.predict(X_test)
```

In [ ]:
```
Y_pred_logistic_reg.shape
```
Out[ ]:
```
(61,)
```

In [ ]:
```
print("Predicted Values : ",Y_pred_logistic_reg)
```
```
Predicted Values :  [0 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 1 1 1 0 1 1 1 1 0
 1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1]
```

In [ ]:
```
Y_test[0:10] #You can check accuracy by observing predicted results and test data.
```
Out[ ]:
```
225    0
152    1
228    0
201    0
52     1
245    0
175    0
168    0
223    0
217    0
Name: target, dtype: int64
```

In [ ]:
```
accuracy_score_logistic_reg = round(accuracy_score(Y_pred_logistic_reg,Y_test)*100,2)
print("The accuracy score achieved using Logistic Regression is: "+str(accuracy_score_logistic_reg)+" %")
```
```
The accuracy score achieved using Logistic Regression is: 85.25 %
```

SVM

In [ ]:
```
from sklearn import svm
model_svm = svm.SVC(kernel='linear')
model_svm.fit(X_train, Y_train)
Y_pred_svm = model_svm.predict(X_test)
```

In [ ]:
```
Y_pred_svm.shape
```
Out[ ]:
```
(61,)
```

In [ ]:
```
print("Predicted Values : ",Y_pred_svm)
```
```
Predicted Values :  [0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 1 1 1 0
 1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1]
```

In [ ]:
```
Y_test[0:10] #You can check accuracy by observing predicted results and test data.
```
Out[ ]:
```
225    0
```

```
152    1
228    0
201    0
52     1
245    0
175    0
168    0
223    0
217    0
Name: target, dtype: int64
```

In [ ]:

```python
accuracy_score_svm = round(accuracy_score(Y_pred_svm,Y_test)*100,2)
print("The accuracy score achieved using Linear SVM is: "+str(accuracy_score_svm)+" %")
```

```
The accuracy score achieved using Linear SVM is: 81.97 %
```

K Nearest Neighbors

In [ ]:

```python
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,Y_train)
Y_pred_knn=knn.predict(X_test)
```

In [ ]:

```python
Y_pred_knn.shape
```

Out[ ]:

```
(61,)
```

In [ ]:

```python
print("Predicted Values : ",Y_pred_knn)
```

```
Predicted Values :  [0 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 1 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0
 1 0 1 0 1 1 0 0 1 1 1 1 1 1 0 1 0 1 0 0 1 0 1 0]
```

In [ ]:

```python
Y_test[0:10] #You can check accuracy by observing predicted results and test data.
```

Out[ ]:

```
225    0
152    1
228    0
201    0
52     1
245    0
175    0
168    0
223    0
217    0
Name: target, dtype: int64
```

In [ ]:

```python
accuracy_score_knn = round(accuracy_score(Y_pred_knn,Y_test)*100,2)
print("The accuracy score achieved using KNN is: "+str(accuracy_score_knn)+" %")
```

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

Decision Tree

In [ ]:

```python
from sklearn.tree import DecisionTreeClassifier
max_accuracy = 0
for x in range(200):
    dt = DecisionTreeClassifier(random_state=x)
    dt.fit(X_train,Y_train)
    Y_pred_dt = dt.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_dt,Y_test)*100,2)
    if(current_accuracy>max_accuracy):
        max_accuracy = current_accuracy
        best_x = x

dt = DecisionTreeClassifier(random_state=best_x)
dt.fit(X_train,Y_train)
```

```
Y_pred_dt = dt.predict(X_test)
```

In [ ]:
```
print(Y_pred_dt.shape)
```

```
(61,)
```

In [ ]:
```
print("Predicted Values : ",Y_pred_dt)
```

```
Predicted Values :  [0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 1 1 1 0
 1 0 0 1 0 1 0 1 0 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1]
```

In [ ]:
```
Y_test[0:10] #You can check accuracy by observing predicted results and test data.
```

Out[ ]:
```
225    0
152    1
228    0
201    0
52     1
245    0
175    0
168    0
223    0
217    0
Name: target, dtype: int64
```

In [ ]:
```
accuracy_score_dt = round(accuracy_score(Y_pred_dt,Y_test)*100,2)
print("The accuracy score achieved using Decision Tree is: "+str(accuracy_score_dt)+" %")
```

```
The accuracy score achieved using Decision Tree is: 81.97 %
```

Random Forest

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
max_accuracy = 0
for x in range(2000):
    rf = RandomForestClassifier(random_state=x)
    rf.fit(X_train,Y_train)
    Y_pred_rf = rf.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_rf,Y_test)*100,2)
    if(current_accuracy>max_accuracy):
        max_accuracy = current_accuracy
        best_x = x

rf = RandomForestClassifier(random_state=best_x)
rf.fit(X_train,Y_train)
Y_pred_rf = rf.predict(X_test)
```

In [ ]:
```
Y_pred_rf.shape
```

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

In [ ]:
```
print("Predicted Values : ",Y_pred_rf)
```

```
Predicted Values :  [0 1 1 0 0 1 0 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0
 1 0 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1]
```

In [ ]:
```
Y_test[0:10] #You can check accuracy by observing predicted results and test data.
```

Out[ ]:
```
225    0
152    1
228    0
201    0
52     1
245    0
175    0
```

```
168    0
223    0
217    0
Name: target, dtype: int64
```

In [ ]:

```
accuracy_score_rf = round(accuracy_score(Y_pred_rf,Y_test)*100,2)
print("The accuracy score achieved using Random Forest is: "+str(accuracy_score_rf)+" %")
```

```
The accuracy score achieved using Random Forest is: 90.16 %
```

Summary of accuracy scores

In [ ]:

```
all_accuracy_scores = [accuracy_score_logistic_reg,accuracy_score_svm,accuracy_score_knn,accuracy_score_dt,accura
algorithms_used = ["Logistic Regression","Support Vector Machine","K-Nearest Neighbors","Decision Tree","Random F
```

In [ ]:

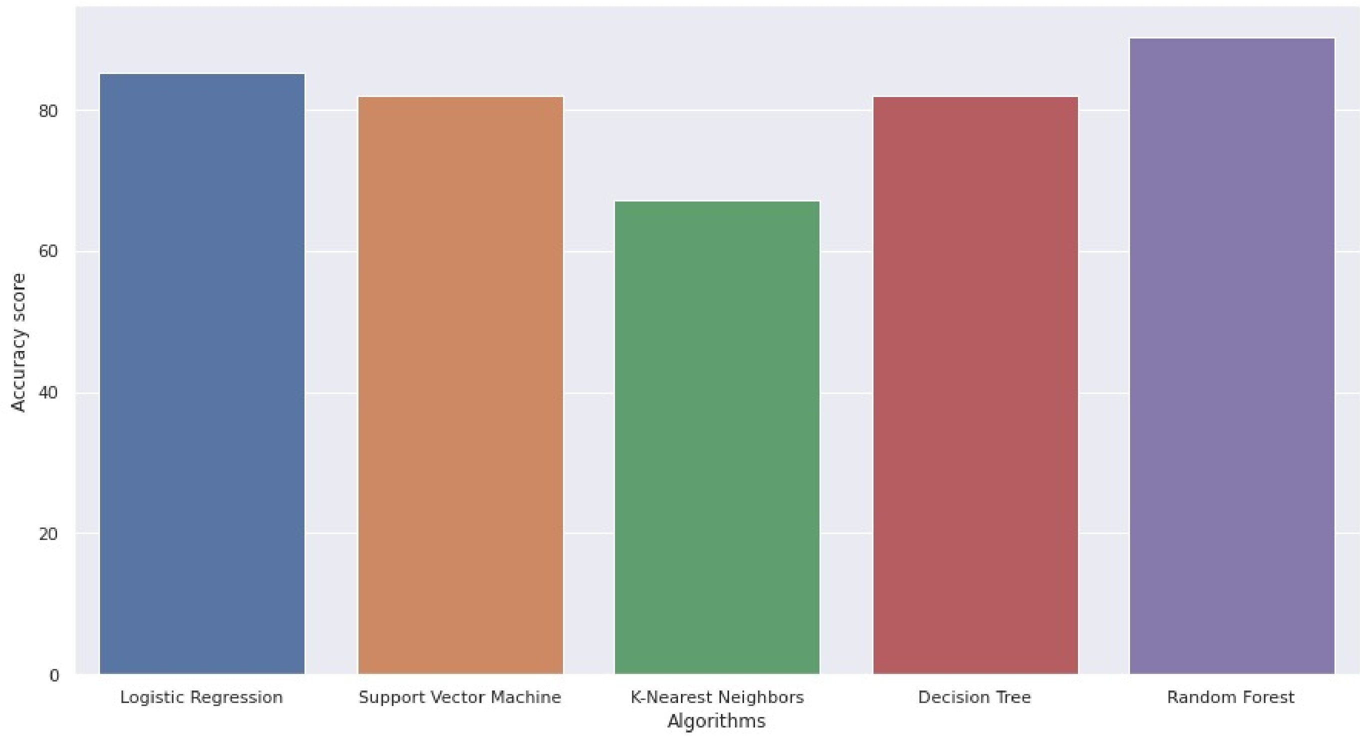
```
for i in range(len(algorithms_used)):
    print("\nThe accuracy score achieved using "+algorithms_used[i]+" is: "+str(all_accuracy_scores[i])+" %")
```

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

The accuracy score achieved using Support Vector Machine is: 81.97 %

The accuracy score achieved using K-Nearest Neighbors is: 67.21 %

The accuracy score achieved using Decision Tree is: 81.97 %

The accuracy score achieved using Random Forest is: 90.16 %
```

In [ ]:

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

sns.barplot(algorithms_used,all_accuracy_scores)
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f898ac16590>



Here we can see that Random Forest is better than other algorithms.