Minor Project - Heart Disease Prediction using Machine Learning

Importing the required libraries

In [1]:
```
import numpy as np # to work for arrays, linear algebra,fourier transformation
import pandas as pd # data analysis and manipulation of tabular data
import matplotlib.pyplot as plt #for plting the graphs
import seaborn as sns #for making stastical graphs

%matplotlib inline

import os
print(os.listdir())

import warnings
warnings.filterwarnings('ignore')
```

```
['.ipynb_checkpoints', '2_minor_proj_ppt.pptx', 'archive.zip', 'Decision Tree a
nd random forest.png', 'Decision Tree.png', 'edit1.ipynb', 'edit1.pdf', 'edit2.
ipynb', 'edit2.pdf', 'edit3.ipynb', 'edit3.pdf', 'final_minor_report.docx', 'fl
ow chart.jpg', 'Heart-Disease-Prediction-using-Machine-Learning', 'heart-diseas
e-prediction-using-machine-learning-IJERTV9IS04061420200510-84272-1pfgh18-with-
cover-page-v2.pdf', 'heart.csv', 'Heart_Disease_Prediction_using_Machine_L.pd
f', 'Heart_Disease_Prediction_using_Machine_L_1.pdf', 'Heart_Disease_Prediction
_using_Machine_L_2.pdf', 'heart_disease_uci.csv', 'knn.png', 'logistic.jpeg',
'Manish_Bhatt_2451137_ProjectIV.docx', 'Manish_Bhatt_2451137_ProjectIV.pdf', 'M
ini_Project_Report_On_Heart_Disease_Pre(1).pdf', 'MINOR PROJECT REPORT.docx',
'MINOR PROJECT REPORT.pdf', 'Minor_Project_Report_2.pdf', 'minor_proj_ppt.ppt
x', 'MP_1.ipynb', 'mp_1.pdf', 'Naive Bayes.png', 'ProposalHeartDeseasePredictio
nSystem.pdf', 'Random Forest.png', 'rp2.pdf', 'svm.png', 'XG Boost.png', 'XG Bo
ost.ppm', '~$2_minor_proj_ppt.pptx']
```

Importing the Dataset

In [2]:
```
data = pd.read_csv(r"C:\Users\Asus\Desktop\Rohan\Semester 5\Minor Proj\heart.csv'
```

In [3]:
```
type(data)
```

Out[3]:
```
pandas.core.frame.DataFrame
```

In [4]:
```
data.shape
```

Out[4]:
```
(303, 14)
```

In [5]:
```
data.head(5)
```

Out[5]:

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

In [6]:
```
data.sample(5)
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slo
114	55	1	1	130	262	0	1	155	0	0.0	
247	66	1	1	160	246	0	1	120	1	0.0	
182	61	0	0	130	330	0	0	169	0	0.0	
213	61	0	0	145	307	0	0	146	1	1.0	
83	52	1	3	152	298	1	1	178	0	1.2	

In [7]:
```
data.describe()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.64686
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.90516
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.00000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.50000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.00000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.00000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.00000

In [8]:
```
data.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
```

Understanding the columns of the dataset

In [9]:
```
info = ["age","1: male, 0: female","chest pain type, 1: typical angina, 2: atypic



for i in range(len(info)):
    print(data.columns[i]+":\t\t\t"+info[i])
```

```
age:                    age
sex:                    1: male, 0: female
cp:                     chest pain type, 1: typical angina, 2: atypical angina,
3: non-anginal pain, 4: asymptomatic
trestbps:                       resting blood pressure
chol:                    serum cholestoral in mg/dl
fbs:                    fasting blood sugar > 120 mg/dl
restecg:                        resting electrocardiographic results (values 0,
1,2)
thalach:                         maximum heart rate achieved
exang:                  exercise induced angina
oldpeak:                        oldpeak = ST depression induced by exercise rel
ative to rest
slope:                  the slope of the peak exercise ST segment
ca:                     number of major vessels (0-3) colored by flourosopy
thal:                   thal: 3 = normal; 6 = fixed defect; 7 = reversable defe
ct
```

In [10]:
```
data['target'].describe()
```

Out[10]:
```
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 [11]:
```
data["target"].unique()
```

Out[11]:
```
array([1, 0], dtype=int64)
```

So this is a classification problem with the target variable having values '0' and '1'

Now checking correlation between columns

In [12]:
```
print(data.corr()["target"].abs().sort_values(ascending=False))
```

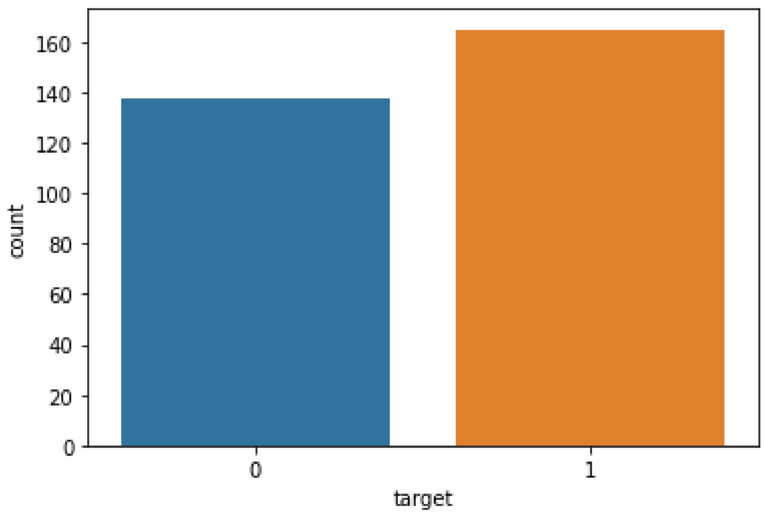
```
target      1.000000
exang       0.436757
cp          0.433798
oldpeak     0.430696
thalach     0.421741
ca          0.391724
slope       0.345877
thal        0.344029
sex         0.280937
age         0.225439
trestbps    0.144931
restecg     0.137230
chol        0.085239
fbs         0.028046
Name: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

Exploratory Data Analysis (EDA)

first analysing the target variable

In [13]:
```
y = data["target"]

sns.countplot(y)


target_temp = data.target.value_counts()

print(target_temp)
```

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



In [14]:
```
print("Percentage of patience without heart problems: "+str(round(target_temp[0]*
print("Percentage of patience with heart problems: "+str(round(target_temp[1]*100
```

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

Now we will analyse 'sex', 'cp', 'fbs'', 'restecg', exang', 'slope', 'ca', and 'thal' feactures

Analysing the 'Sex' feacture

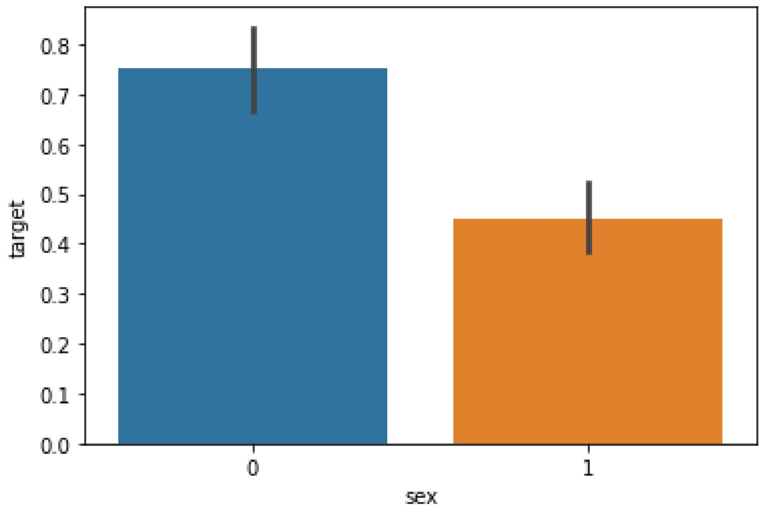
In [15]:
```
data["sex"].unique()
```

Out[15]:
```
array([1, 0], dtype=int64)
```

As expected, the 'sex' feacture has 2 uinque feactures

In [16]:
```
# now plotting the sex ratio
sns.barplot(data["sex"],y)
```

Out[16]: <AxesSubplot:xlabel='sex', ylabel='target'>



Analysing the 'Chest Pain Type' feacture

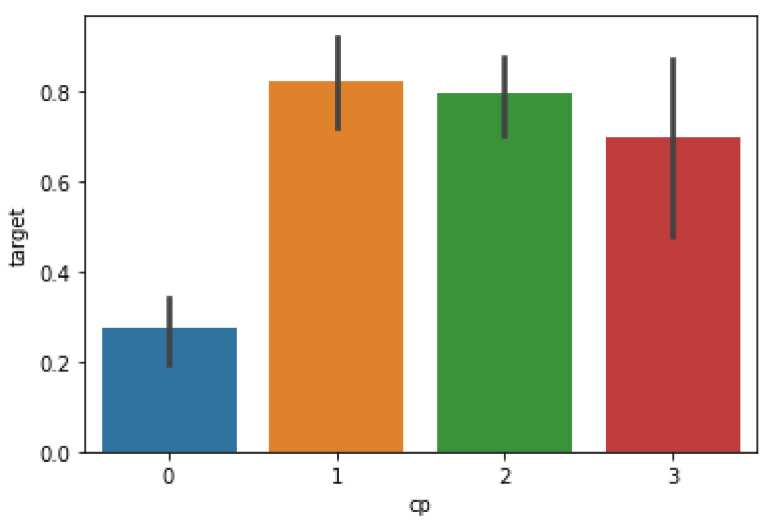
In [17]:
```
data["cp"].unique()
```

Out[17]: array([3, 2, 1, 0], dtype=int64)

In [18]:
```
#the cp feacture has values from 0 to 3
```

In [19]:
```
#plotting the graph
sns.barplot(data["cp"],y)
```

Out[19]: <AxesSubplot:xlabel='cp', ylabel='target'>



Analysing the 'Fasting Blood Sugar (FBS)' feacture

In [20]:
```
data["fbs"].describe()
```

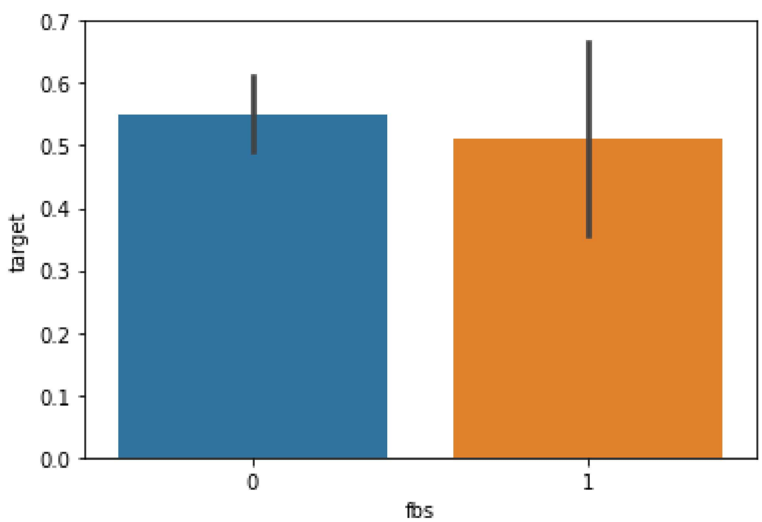
Out[20]:
```
count    303.000000
mean       0.148515
std        0.356198
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000
Name: fbs, dtype: float64
```

In [21]:
```
data["fbs"].unique()
```

Out[21]:
```
array([1, 0], dtype=int64)
```

In [22]:
```
#plotting the graph
sns.barplot(data["fbs"],y)
```

Out[22]:
```
<AxesSubplot:xlabel='fbs', ylabel='target'>
```



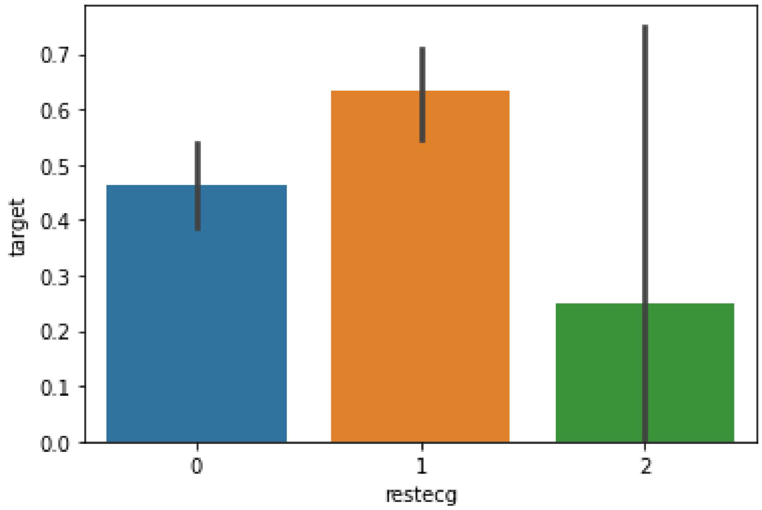
Analysing the 'Resting Electro Cardiographic Results (restecg)' feacture

In [23]:
```
data["restecg"].unique()
```

Out[23]:
```
array([0, 1, 2], dtype=int64)
```

In [24]:
```
sns.barplot(data["restecg"],y)
```

Out[24]: <AxesSubplot:xlabel='restecg', ylabel='target'>



In [25]:
```
# we find out that the people with the restecg '1' and '0'
# are much more likely to have a heart disease than with restecg '2'
```

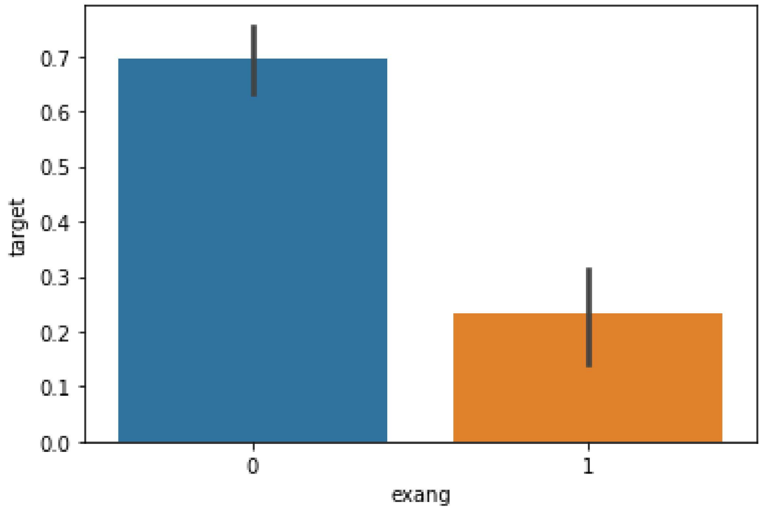
Analysing the 'Exercise - induced angina (exang)' feacture

In [26]:
```
data["exang"].unique()
```

Out[26]: array([0, 1], dtype=int64)

In [27]:
```
sns.barplot(data["exang"],y)
```

Out[27]: <AxesSubplot:xlabel='exang', ylabel='target'>



In [28]:
```
#people with exang =1 i.e. Exercise included angina are
# much less likely to have heart problems
```

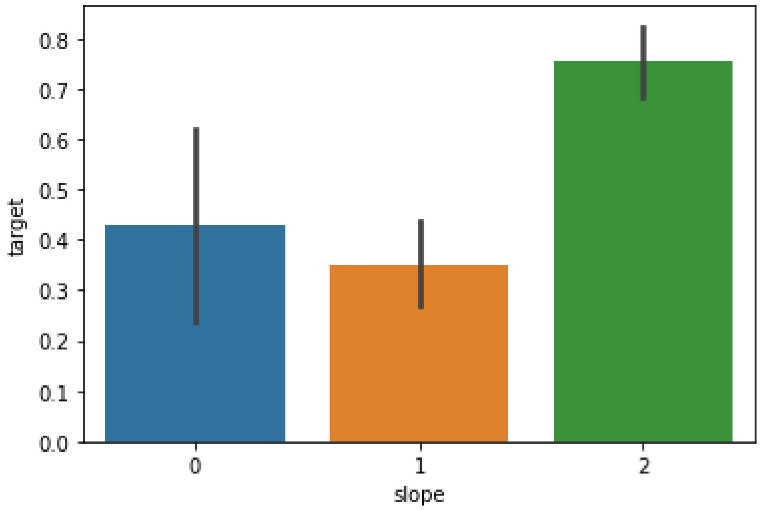
Analysing the 'Slope (slope of the peak exericse)' feacture

In [29]:
```
data["slope"].unique()
```

Out[29]: array([0, 2, 1], dtype=int64)

In [30]:
```
sns.barplot(data["slope"],y)
```

Out[30]: <AxesSubplot:xlabel='slope', ylabel='target'>



In [31]:
```
# So, slope '2' causes heart pain much more than
# slope '0' and '1'
```

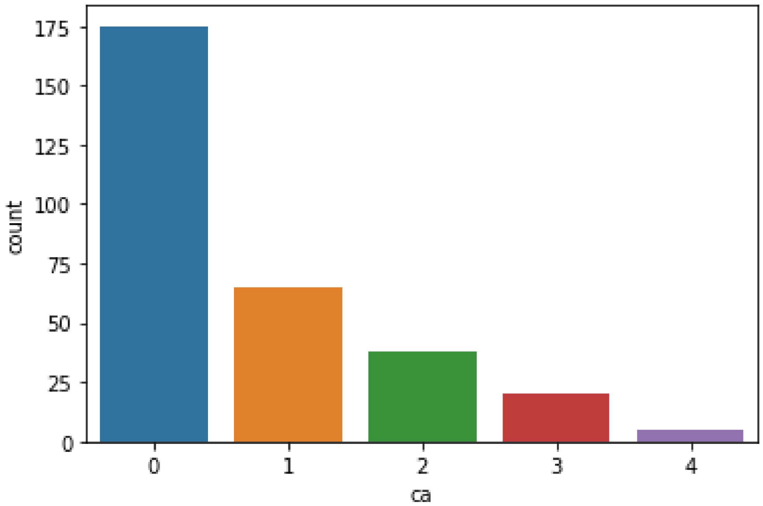
Analysing the 'ca (no of major vessels)' feacture

In [32]:
```
data["ca"].unique()
```

Out[32]: array([0, 2, 1, 3, 4], dtype=int64)

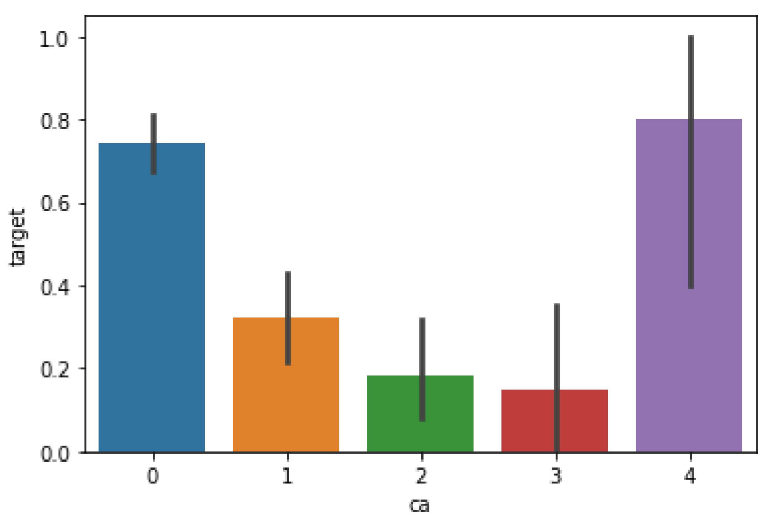
In [33]:
```
sns.countplot(data["ca"])
```

Out[33]: <AxesSubplot:xlabel='ca', ylabel='count'>



In [34]:
```
sns.barplot(data["ca"],y)
```

Out[34]: <AxesSubplot:xlabel='ca', ylabel='target'>



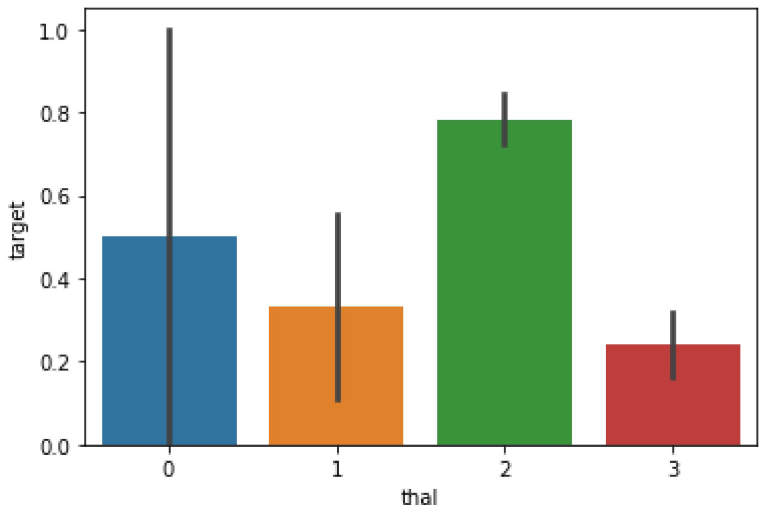
Analysing the 'thal' feacture

In [35]:
```
data["thal"].unique()
```

Out[35]:
```
array([1, 2, 3, 0], dtype=int64)
```

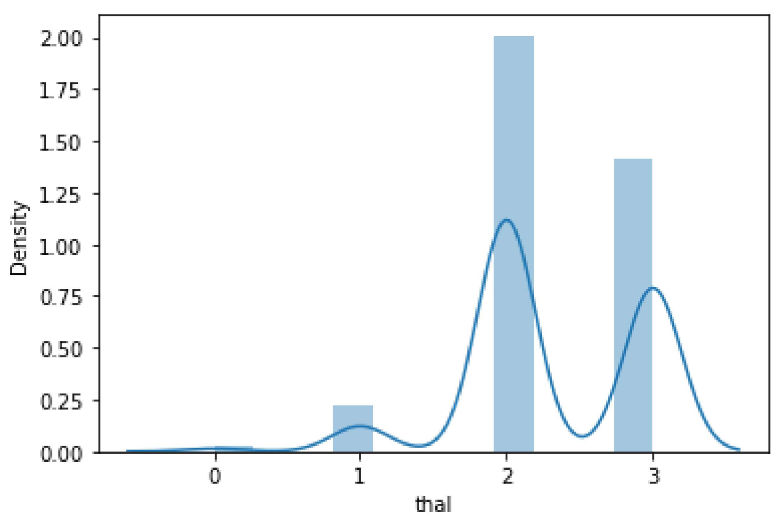
In [36]:
```
sns.barplot(data["thal"],y)
```

Out[36]:
```
<AxesSubplot:xlabel='thal', ylabel='target'>
```



In [37]:
```
sns.distplot(data["thal"])
```

Out[37]:
```
<AxesSubplot:xlabel='thal', ylabel='Density'>
```



Training and Testing our model

In [38]:
```
from sklearn.model_selection import train_test_split

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
```

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,)

V Model fitting

In [43]:
```
from sklearn.metrics import accuracy_score
```

Logistic Regression

In [44]:
```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()

lr.fit(X_train,Y_train)

Y_pred_lr = lr.predict(X_test)
```

In [45]:
```
Y_pred_lr.shape
```

Out[45]: (61,)

In [46]:
```
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 %
```

Naive Bayes

In [47]:
```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()

nb.fit(X_train,Y_train)

Y_pred_nb = nb.predict(X_test)
```

In [48]:
```
Y_pred_nb.shape
```

Out[48]:
```
(61,)
```

In [49]:
```
score_nb = round(accuracy_score(Y_pred_nb,Y_test)*100,2)

print("The accuracy score achieved using Naive Bayes is: "+str(score_nb)+" %")
```

```
The accuracy score achieved using Naive Bayes is: 85.25 %
```

Support Vector Machine (SVM)

In [50]:
```
from sklearn import svm

sv = svm.SVC(kernel='linear')

sv.fit(X_train, Y_train)

Y_pred_svm = sv.predict(X_test)
```

In [51]:
```
Y_pred_svm.shape
```

Out[51]:
```
(61,)
```

In [52]:
```
score_svm = round(accuracy_score(Y_pred_svm,Y_test)*100,2)

print("The accuracy score achieved using Linear SVM is: "+str(score_svm)+" %")
```

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

K Nearest Neighbors (KNN)

In [53]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,Y_train)
Y_pred_knn=knn.predict(X_test)
```

In [54]:
```
Y_pred_knn.shape
```

Out[54]:
```
(61,)
```

In [55]:
```
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 %
```

Decision Tree

In [56]:
```
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

#print(max_accuracy)
#print(best_x)


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

In [57]:
```
print(Y_pred_dt.shape)
```

```
(61,)
```

In [58]:
```
score_dt = round(accuracy_score(Y_pred_dt,Y_test)*100,2)

print("The accuracy score achieved using Decision Tree is: "+str(score_dt)+" %")
```

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

Random Forest

In [59]:
```
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

#print(max_accuracy)
#print(best_x)

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

In [60]:
```
Y_pred_rf.shape
```

Out[60]: (61,)

In [61]:
```
score_rf = round(accuracy_score(Y_pred_rf,Y_test)*100,2)

print("The accuracy score achieved using Decision Tree is: "+str(score_rf)+" %")
```

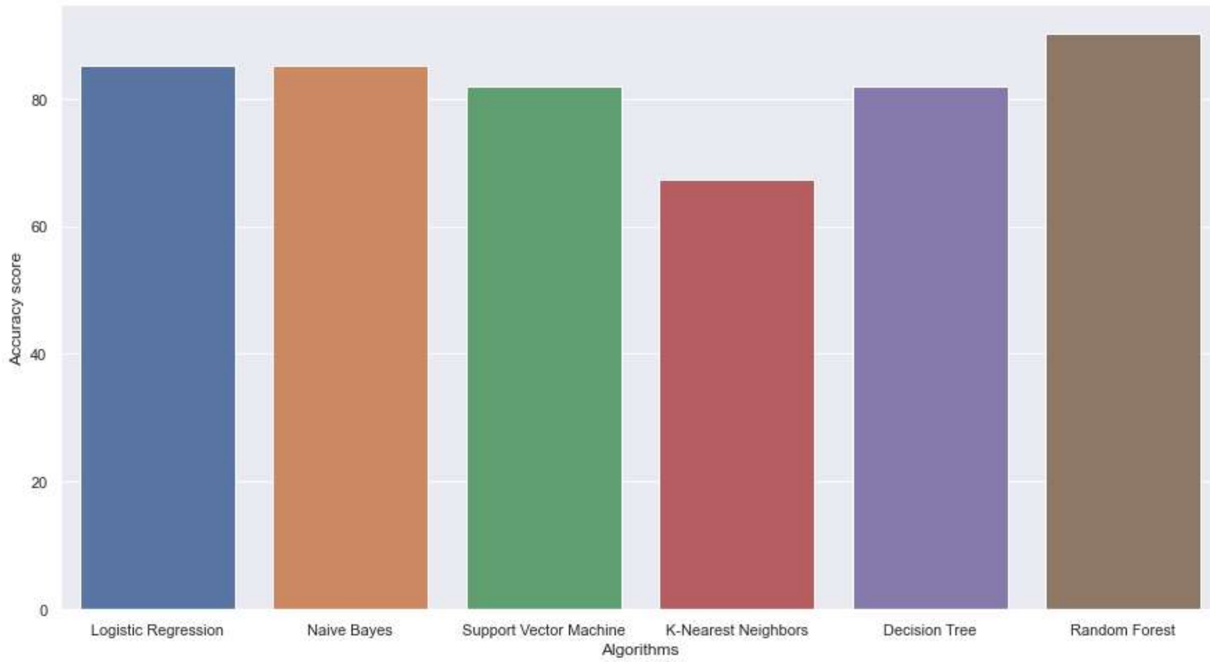
```
The accuracy score achieved using Decision Tree is: 90.16 %
```

In [65]:
```
scores = [score_lr,score_nb,score_svm,score_knn,score_dt,score_rf]
algorithms = ["Logistic Regression","Naive Bayes","Support Vector Machine","K-Nea

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

```
The accuracy score achieved using Logistic Regression is: 85.25 %
The accuracy score achieved using Naive Bayes 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 [66]:
```
sns.set(rc={'figure.figsize':(15,8)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

Out[66]:
```
<AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>
```



In [67]:
```
import pickle
```

In [72]:
```
filename='heart_pred_model'
pickle.dump(rf,open(filename,'wb'))
```

In [75]:
```
loadedmodel = pickle.load(open(filename,'rb'))
loadedmodel.predict(X_test)
```

Out[75]:
```
array([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, 0, 0, 0, 1, 1, 1, 0, 1, 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], dtype=int64)
```

In [ ]:
```

```