Import Libraries

In [56]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

import os
print(os.listdir())


```

```
['.anaconda', '.android', '.astropy', '.conda', '.condarc', '.continuum', '.docker', '.emulator_con
sole_auth_token', '.gradle', '.idea', '.ipynb_checkpoints', '.ipython', '.jdks', '.jupyter', '.kera
s', '.m2', '.matplotlib', '.ms-ad', '.vscode', '1finds.csv', '2ce.csv', '3D Objects', '5user_data.c
sv', 'algorithm.ipynb', 'anaconda3', 'AndroidStudioProjects', 'Another image.ipynb', 'App.py.ipyn
b', 'AppData', 'Application Data', 'Contacts', 'Cookies', 'Customer Churn Prediction.ipynb', 'custo
mer_churn_large_dataset.csv', 'customer_churn_large_dataset.xlsx', 'dask-worker-space', 'Data prepr
ocessing.ipynb', 'datasets', 'day 3.ipynb', 'decision tree.ipynb', 'Desktop', 'Diabetes prediction
system.ipynb', 'Diabetes prediction.ipynb', 'diabetes.csv', 'Diabetes.ipynb', 'DiabetesPrection',
'DiabetesPrediction', 'Digital image processing.ipynb', 'Documents', 'Downloads', 'End to End proje
ct.ipynb', 'Favorites', 'Gagana.ipynb', 'Heart Disease Prediction using Maching Learning.ipynb', 'H
eart Disease Prediction.ipynb', 'heart.csv', 'housing.csv', 'IntelGraphicsProfiles', 'Internship Pr
oject.ipynb', 'iris.csv', 'java_error_in_studio64.hprof', 'KNN.ipynb', 'Links', 'Local Settings',
'Music', 'My Documents', 'naivetext.csv', 'NetHood', 'non linear svm.ipynb', 'NTUSER.DAT', 'ntuser.
dat.LOG1', 'ntuser.dat.LOG2', 'NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TM.blf', 'NTUSER.DA
T{53b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer00000000000000000001.regtrans-ms', 'NTUSER.DAT{5
3b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer00000000000000000002.regtrans-ms', 'ntuser.ini', 'o
il_spill.csv', 'OneDrive', 'page-title15Mars3.csv', 'Pictures', 'pima_indian.csv', 'pizzaplace.cs
v', 'pizza_price_predict', 'plots.ipynb', 'Practice Program.ipynb', 'Prinston Internship project.ip
ynb', 'Prinston project.ipynb', 'PrintHood', 'python', 'Recent', 'Saved Games', 'Searches', 'SendT
o', 'server', 'Start Menu', 'svm.ipynb', 'Templates', 'test31MarsFINAL.xlsx', 'todo_app', 'trained_
model.sav', 'training_model.sav', 'Untitled.ipynb', 'Untitled1.ipynb', 'Untitled10.ipynb', 'Untitle
d11.ipynb', 'Untitled12.ipynb', 'Untitled13.ipynb', 'Untitled14.ipynb', 'Untitled15.ipynb', 'Untitl
ed16.ipynb', 'Untitled17.ipynb', 'Untitled18.ipynb', 'Untitled19.ipynb', 'Untitled2.ipynb', 'Untitl
ed20.ipynb', 'Untitled21.ipynb', 'Untitled22.ipynb', 'Untitled23.ipynb', 'Untitled24.ipynb', 'Untit
led25.ipynb', 'Untitled26.ipynb', 'Untitled27.ipynb', 'Untitled28.ipynb', 'Untitled29.ipynb', 'Unti
tled3.ipynb', 'Untitled30.ipynb', 'Untitled31.ipynb', 'Untitled32.ipynb', 'Untitled33.ipynb', 'Unti
tled34.ipynb', 'Untitled35.ipynb', 'Untitled36.ipynb', 'Untitled37.ipynb', 'Untitled38.ipynb', 'Unt
itled39.ipynb', 'Untitled4.ipynb', 'Untitled40.ipynb', 'Untitled41.ipynb', 'Untitled42.ipynb', 'Unt
itled43.ipynb', 'Untitled44.ipynb', 'Untitled45.ipynb', 'Untitled46.ipynb', 'Untitled47.ipynb', 'Un
titled48.ipynb', 'Untitled49.ipynb', 'Untitled5.ipynb', 'Untitled50.ipynb', 'Untitled51.ipynb', 'Un
titled52.ipynb', 'Untitled53.ipynb', 'Untitled54.ipynb', 'Untitled55.ipynb', 'Untitled56.ipynb', 'U
ntitled57.ipynb', 'Untitled58.ipynb', 'Untitled59.ipynb', 'Untitled6 (1).ipynb', 'Untitled6.ipynb',
'Untitled60.ipynb', 'Untitled61.ipynb', 'Untitled62.ipynb', 'Untitled63.ipynb', 'Untitled64.ipynb',
'Untitled65.ipynb', 'Untitled66.ipynb', 'Untitled7.ipynb', 'Untitled8.ipynb', 'Untitled9.ipynb', 'V
ideos', 'weather.csv']
```

Importing and Understanding our Dataset

In [57]:
```
dataset = pd.read_csv("heart.csv")
```

In [58]:
```
type(dataset)

```

Out[58]:
```
pandas.core.frame.DataFrame
```

In [59]:
```
dataset.shape

```

Out[59]:
```
(1025, 14)
```

In [60]:
```
1 dataset.head(5)
```

Out[60]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

In [61]:
```
1 dataset.sample(5)
```

Out[61]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
632	45	0	1	130	234	0	0	175	0	0.6	1	0	2	1
984	59	1	0	135	234	0	1	161	0	0.5	1	0	3	1
327	57	1	0	150	276	0	0	112	1	0.6	1	1	1	0
697	67	1	2	152	212	0	0	150	0	0.8	1	0	3	0
637	58	0	2	120	340	0	1	172	0	0.0	2	0	2	1

In [62]:
```
1 dataset.describe()
```

Out[62]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000

In [63]:
```
1 dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [64]:
```
info = ["age","1: male, 0: female","chest pain type, 1: typical angina, 2: atypical angina, 3: n

for i in range(len(info)):
    print(dataset.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 pai
n, 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 relative 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 defect
```

In [65]:
```
dataset["target"].describe()
```

Out[65]:
```
count    1025.000000
mean        0.513171
std         0.500070
min         0.000000
25%         0.000000
50%         1.000000
75%         1.000000
max         1.000000
Name: target, dtype: float64
```

In [66]:
```
dataset["target"].unique()
```

Out[66]:
```
array([0, 1], dtype=int64)
```

Checking correlation between columns

In [67]:
```
print(dataset.corr()["target"].abs().sort_values(ascending=False))
```

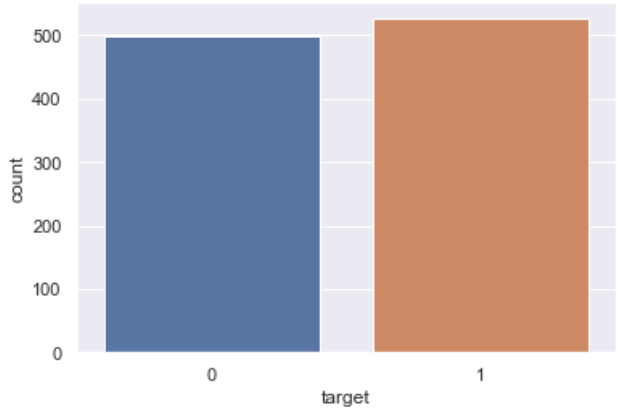
```
target      1.000000
oldpeak     0.438441
exang       0.438029
cp          0.434854
thalach     0.422895
ca          0.382085
slope       0.345512
thal        0.337838
sex         0.279501
age         0.229324
trestbps    0.138772
restecg     0.134468
chol        0.099966
fbs         0.041164
Name: target, dtype: float64
```

Exploratory Data Analysis (EDA)

First, analysing the target variable:

In [68]:
```
y = dataset["target"]

sns.countplot(y)


target_temp = dataset.target.value_counts()

print(target_temp)

```

```
1    526
0    499
Name: target, dtype: int64

C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followin
g variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data
`, and passing other arguments without an explicit keyword will result in an error or misinterpreta
tion.
  warnings.warn(
```



In [69]:
```
print("Percentage of patience without heart problems: "+str(round(target_temp[0]*100/303,2)))
print("Percentage of patience with heart problems: "+str(round(target_temp[1]*100/303,2)))

```

```
Percentage of patience without heart problems: 164.69
Percentage of patience with heart problems: 173.6
```

Analysing the 'Sex' feature

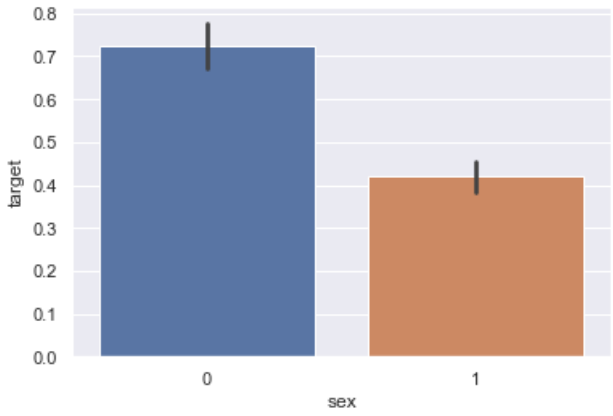
In [70]:
```
dataset["sex"].unique()
```

Out[70]:
```
array([1, 0], dtype=int64)
```

In [71]:
```
1 sns.barplot(dataset["sex"],y)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

Out[71]: <AxesSubplot:xlabel='sex', ylabel='target'>



Analysing the 'Chest Pain Type' feature

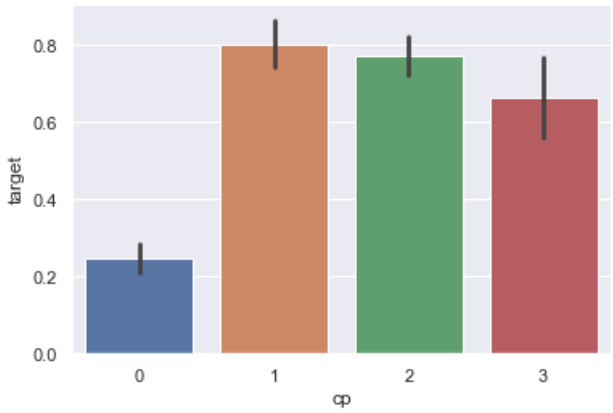
In [72]:
```
1 dataset["cp"].unique()
```

Out[72]: array([0, 1, 2, 3], dtype=int64)

In [73]:
```
1 sns.barplot(dataset["cp"],y)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

Out[73]: <AxesSubplot:xlabel='cp', ylabel='target'>



Analysing the FBS feature

In [74]:
```
dataset["fbs"].describe()

```

Out[74]:
```
count    1025.000000
mean        0.149268
std         0.356527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
Name: fbs, dtype: float64
```

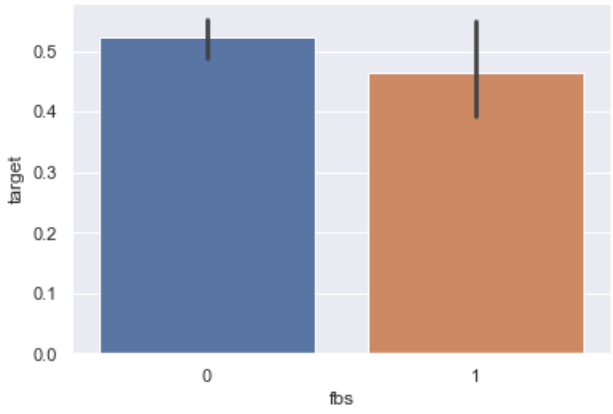
In [75]:
```
dataset["fbs"].unique()

```

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

In [76]:
```
sns.barplot(dataset["fbs"],y)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

Out[76]:
```
<AxesSubplot:xlabel='fbs', ylabel='target'>
```



Analysing the restecg feature

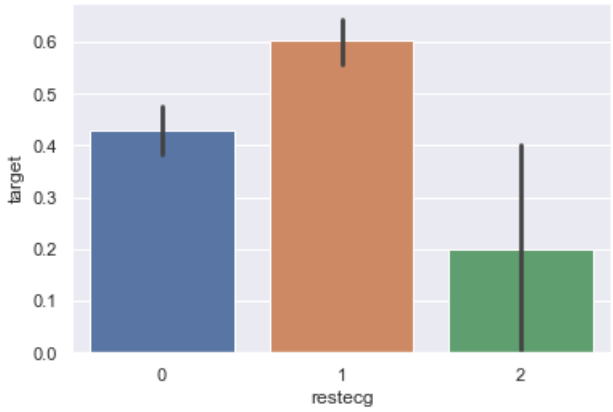
In [77]:
```
dataset["restecg"].unique()
```

Out[77]:
```
array([1, 0, 2], dtype=int64)
```

In [78]:
```
1 sns.barplot(dataset["restecg"],y)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

Out[78]: <AxesSubplot:xlabel='restecg', ylabel='target'>



Analysing the 'exang' feature

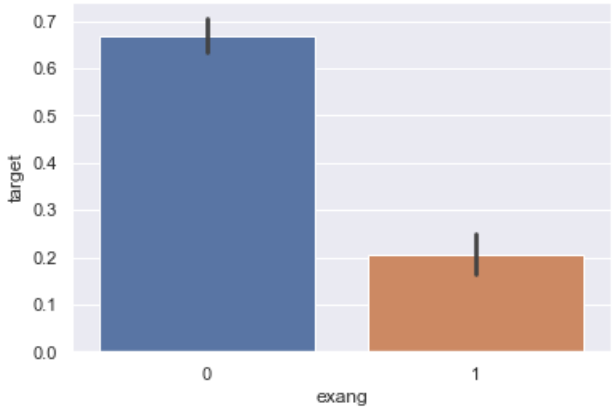
In [79]:
```
1 dataset["exang"].unique()
```

Out[79]: array([0, 1], dtype=int64)

In [80]:
```
1 sns.barplot(dataset["exang"],y)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

Out[80]: <AxesSubplot:xlabel='exang', ylabel='target'>



Analysing the Slope feature

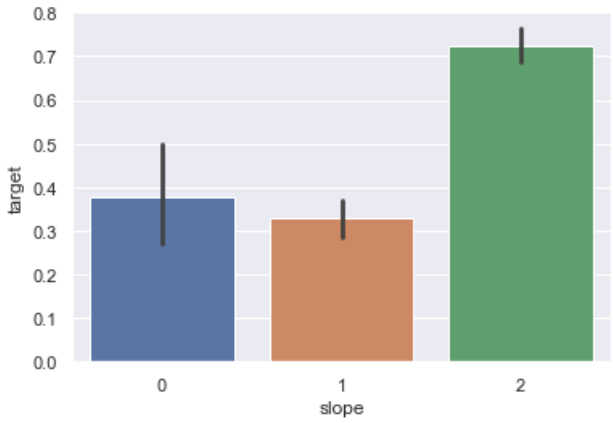
In [81]:
```
1 dataset["slope"].unique()
```

Out[81]: array([2, 0, 1], dtype=int64)

In [82]:
```
1 sns.barplot(dataset["slope"],y)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

Out[82]: <AxesSubplot:xlabel='slope', ylabel='target'>



Analysing the 'ca' feature

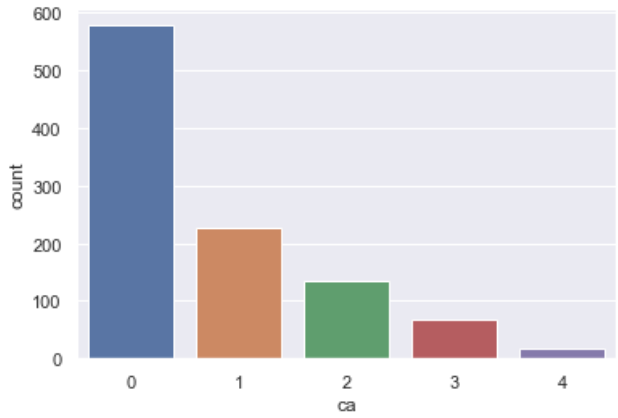
In [83]:
```
1 dataset["ca"].unique()
```

Out[83]: array([2, 0, 1, 3, 4], dtype=int64)

In [84]:
```
1 sns.countplot(dataset["ca"])
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followin
g variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data
`, and passing other arguments without an explicit keyword will result in an error or misinterpreta
tion.
  warnings.warn(
```

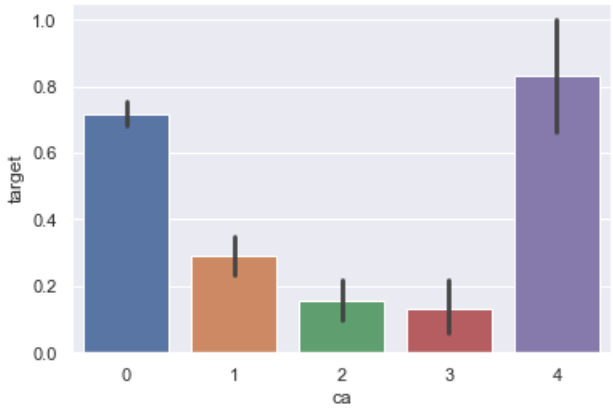
Out[84]: <AxesSubplot:xlabel='ca', ylabel='count'>



In [85]:
```
1 sns.barplot(dataset["ca"],y)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

Out[85]: <AxesSubplot:xlabel='ca', ylabel='target'>



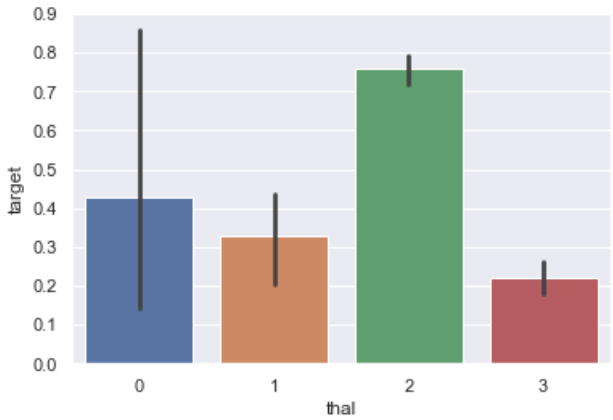
In [86]:
```
1 dataset["thal"].unique()
```

Out[86]: array([3, 2, 1, 0], dtype=int64)

In [87]:
```
1 sns.barplot(dataset["thal"],y)
2
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

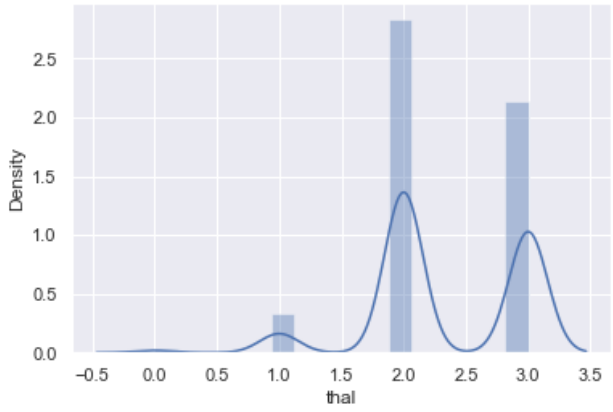
Out[87]: <AxesSubplot:xlabel='thal', ylabel='target'>



In [88]:
```
1 sns.distplot(dataset["thal"])
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either
`displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function
for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[88]: <AxesSubplot:xlabel='thal', ylabel='Density'>



Train Test split

In [89]:
```
1 from sklearn.model_selection import train_test_split
2
3 predictors = dataset.drop("target",axis=1)
4 target = dataset["target"]
5
6 X_train,X_test,Y_train,Y_test = train_test_split(predictors,target,test_size=0.20,random_state=0
```

In [90]:
```
1 X_train.shape
```

Out[90]: (820, 13)

In [91]:
```
1 X_test.shape
```

Out[91]: (205, 13)

In [92]:
```
1 Y_train.shape
```

Out[92]: (820,)

In [93]:
```
1 Y_test.shape
```

Out[93]: (205,)

Model Fitting

In [94]:
```
1 from sklearn.metrics import accuracy_score
```

In [95]:
```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()

lr.fit(X_train,Y_train)

Y_pred_lr = lr.predict(X_test)
Y_pred_lr.shape
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[95]:
```
(205,)
```

In [96]:
```
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: 86.34 %
```

Naive Bayes

In [97]:
```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()

nb.fit(X_train,Y_train)

Y_pred_nb = nb.predict(X_test)
Y_pred_nb.shape
```

Out[97]:
```
(205,)
```

In [98]:
```
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: 85.37 %
```

SVM

In [99]:
```
from sklearn import svm

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

sv.fit(X_train, Y_train)

Y_pred_svm = sv.predict(X_test)
Y_pred_svm.shape
```

Out[99]:
```
(205,)
```

In [100]:
```
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: 83.9 %
```

K Nearest Neighbors

In [101]:
```
from sklearn.neighbors import KNeighborsClassifier

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

Out[101]: (205,)

In [102]:
```
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: 72.2 %
```

Decision Tree

In [103]:
```
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 [104]:
```
print(Y_pred_dt.shape)
```

```
(205,)
```

In [105]:
```
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: 100.0 %
```

XGBoost

In [106]:
```
import xgboost as xgb

xgb_model = xgb.XGBClassifier(objective="binary:logistic", random_state=42)
xgb_model.fit(X_train, Y_train)

Y_pred_xgb = xgb_model.predict(X_test)
```

In [107]:
```
Y_pred_xgb.shape
```

Out[107]: (205,)

In [108]:
```
score_xgb = round(accuracy_score(Y_pred_xgb,Y_test)*100,2)

print("The accuracy score achieved using XGBoost is: "+str(score_xgb)+" %")
```

```
The accuracy score achieved using XGBoost is: 100.0 %
```

Output final score

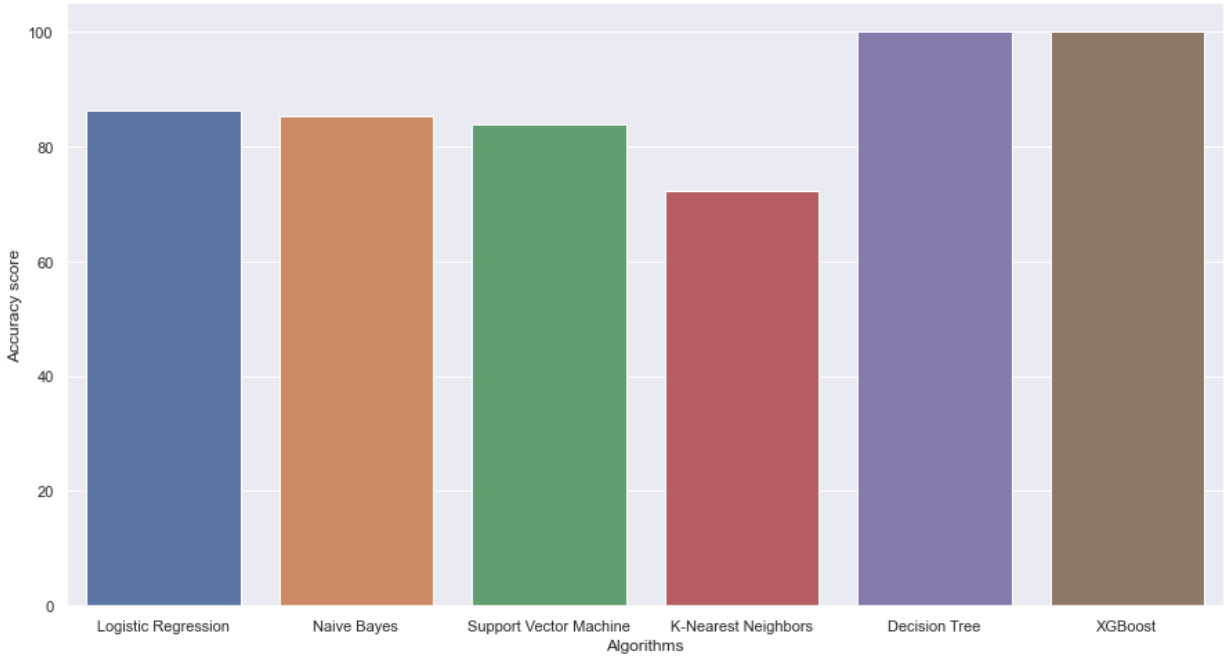
In [109]:
```
scores = [score_lr,score_nb,score_svm,score_knn,score_dt,score_xgb]
algorithms = ["Logistic Regression","Naive Bayes","Support Vector Machine","K-Nearest Neighbors","De

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: 86.34 %
The accuracy score achieved using Naive Bayes is: 85.37 %
The accuracy score achieved using Support Vector Machine is: 83.9 %
The accuracy score achieved using K-Nearest Neighbors is: 72.2 %
The accuracy score achieved using Decision Tree is: 100.0 %
The accuracy score achieved using XGBoost is: 100.0 %
```

In [110]:
```
sns.set(rc={'figure.figsize':(15,8)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

Out[110]:
```
<AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>
```



Decision Tree and XGBoost has good result as compare to other algorithm

In [ ]:
```
```