Importing Essential Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings. filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn import *

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

Importing Dataset

```
heartData=pd.read_csv(r"C:\Users\navya\Downloads\hearts.csv");

heartData.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
```

More Info About Columns

```
info = ["age","1: male, 0: female","chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic","resting blood pressure"," serum cholestoral in mg/dl","fasting blood sugar > 120 mg/dl","resting electrocardiographic results (values 0,1,2)"," maximum heart rate achieved","exercise induced angina","oldpeak = ST depression induced by exercise relative to rest","the slope of the peak exercise ST segment","number of major vessels (0-3) colored by flourosopy","thal: 3 = normal; 6 = fixed defect; 7 = reversable defect"]



for i in range(len(info)):
    print(heartData.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
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
```

```
heartData
```

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang
oldpeak  \
0      52    1   0       125   212    0        1      168      0
1.0
1      53    1   0       140   203    1        0      155      1
3.1
```

```
2      70    1   0       145   174    0        1      125      1
2.6
3      61    1   0       148   203    0        1      161      0
0.0
4      62    0   0       138   294    1        1      106      0
1.9
...   ...  ...  ..       ...   ...  ...      ...      ...    ...
...
1020   59    1   1       140   221    0        1      164      1
0.0
1021   60    1   0       125   258    0        0      141      1
2.8
1022   47    1   0       110   275    0        0      118      1
1.0
1023   50    0   0       110   254    0        0      159      0
0.0
1024   54    1   0       120   188    0        1      113      0
1.4

      slope  ca  thal  target
0         2   2     3       0
1         0   0     3       0
2         0   0     3       0
3         2   1     3       0
4         1   3     2       0
...     ...  ..   ...     ...
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0

[1025 rows x 14 columns]
```

Description

```
heartData.describe()

               age          sex           cp     trestbps         chol
\
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.00000

mean     54.434146     0.695610     0.942439   131.611707   246.00000

std       9.072290     0.460373     1.029641    17.516718    51.59251

min      29.000000     0.000000     0.000000    94.000000   126.00000
```

```
25%      48.000000     0.000000     0.000000   120.000000   211.00000

50%      56.000000     1.000000     1.000000   130.000000   240.00000

75%      61.000000     1.000000     2.000000   140.000000   275.00000

max      77.000000     1.000000     3.000000   200.000000   564.00000


               fbs      restecg      thalach        exang      oldpeak
\
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000

mean      0.149268     0.529756   149.114146     0.336585     1.071512

std       0.356527     0.527878    23.005724     0.472772     1.175053

min       0.000000     0.000000    71.000000     0.000000     0.000000

25%       0.000000     0.000000   132.000000     0.000000     0.000000

50%       0.000000     1.000000   152.000000     0.000000     0.800000

75%       0.000000     1.000000   166.000000     1.000000     1.800000

max       1.000000     2.000000   202.000000     1.000000     6.200000


             slope           ca         thal       target
count  1025.000000  1025.000000  1025.000000  1025.000000
mean      1.385366     0.754146     2.323902     0.513171
std       0.617755     1.030798     0.620660     0.500070
min       0.000000     0.000000     0.000000     0.000000
25%       1.000000     0.000000     2.000000     0.000000
50%       1.000000     0.000000     2.000000     1.000000
75%       2.000000     1.000000     3.000000     1.000000
max       2.000000     4.000000     3.000000     1.000000
```

Missing Values

```
heartData.isnull().sum()
```

```
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
```

```
missing_data= heartData.isnull().sum()
total_percentage = (missing_data.sum()/heartData.shape[0]) * 100
print(f'Total percentage of missing data is
{round(total_percentage,2)}%')
duplicate=heartData[heartData.duplicated()]
print("Duplicate rows:")
duplicate
#drop duplicate rows
heartData=heartData.drop_duplicates()
```

```
Total percentage of missing data is 0.0%
Duplicate rows:
```

Analysing the 'target variable'

```
heartData["target"].describe()
```

```
count    302.000000
mean       0.543046
std        0.498970
min        0.000000
25%        0.000000
50%        1.000000
75%        1.000000
max        1.000000
Name: target, dtype: float64
```

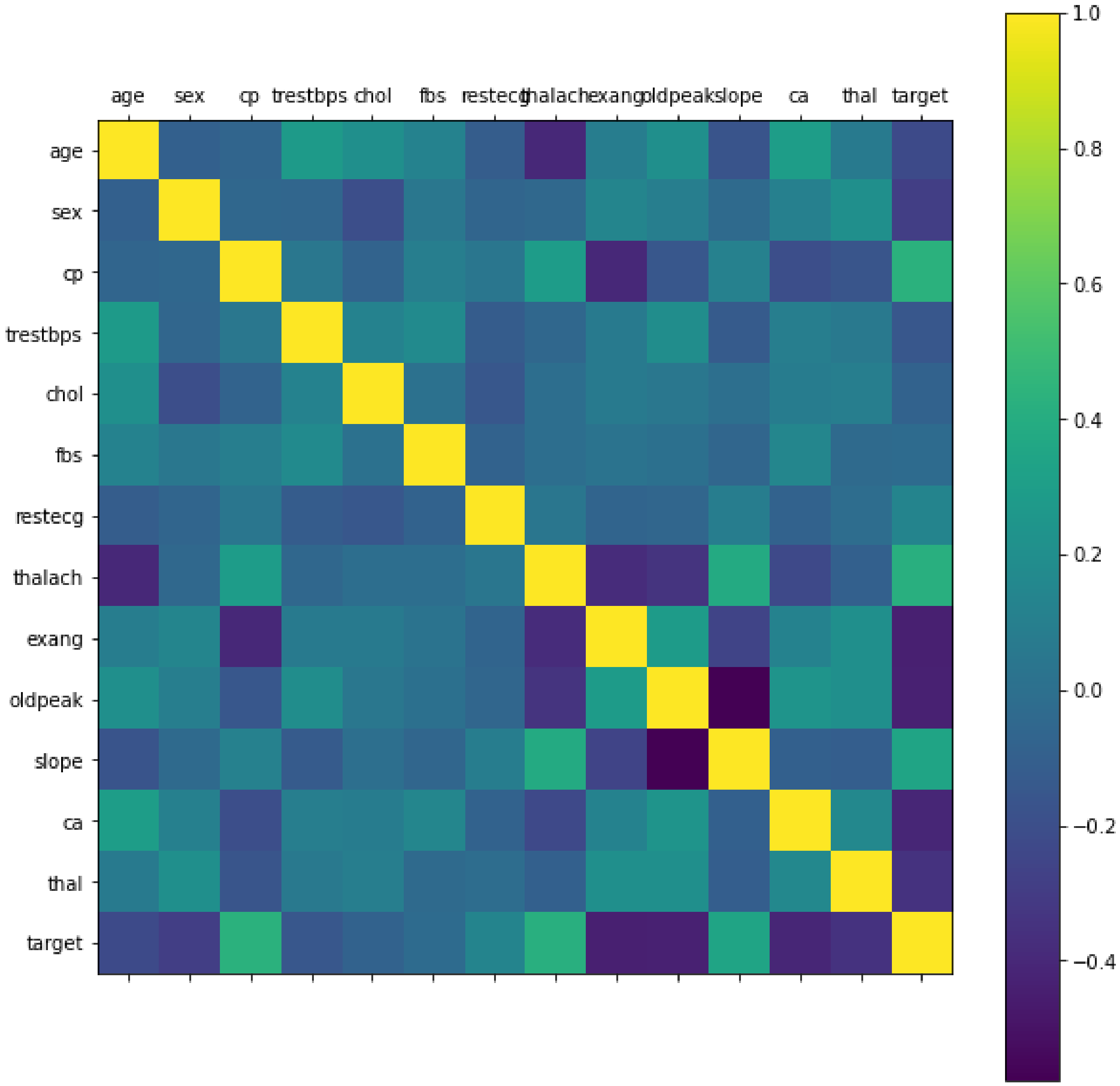
```
print(heartData.corr()["target"].abs().sort_values(ascending=False))
```

```
target      1.000000
exang       0.435601
cp          0.432080
oldpeak     0.429146
thalach     0.419955
ca          0.408992
slope       0.343940
thal        0.343101
sex         0.283609
age         0.221476
trestbps    0.146269
```

```
restecg     0.134874
chol        0.081437
fbs         0.026826
Name: target, dtype: float64
```

```
rcParams['figure.figsize'] = 10,10
plt.matshow(heartData.corr())
plt.yticks(np.arange(heartData.shape[1]), heartData.columns)
plt.xticks(np.arange(heartData.shape[1]), heartData.columns)
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x1d5bf82afd0>
```



checking correlation between columns

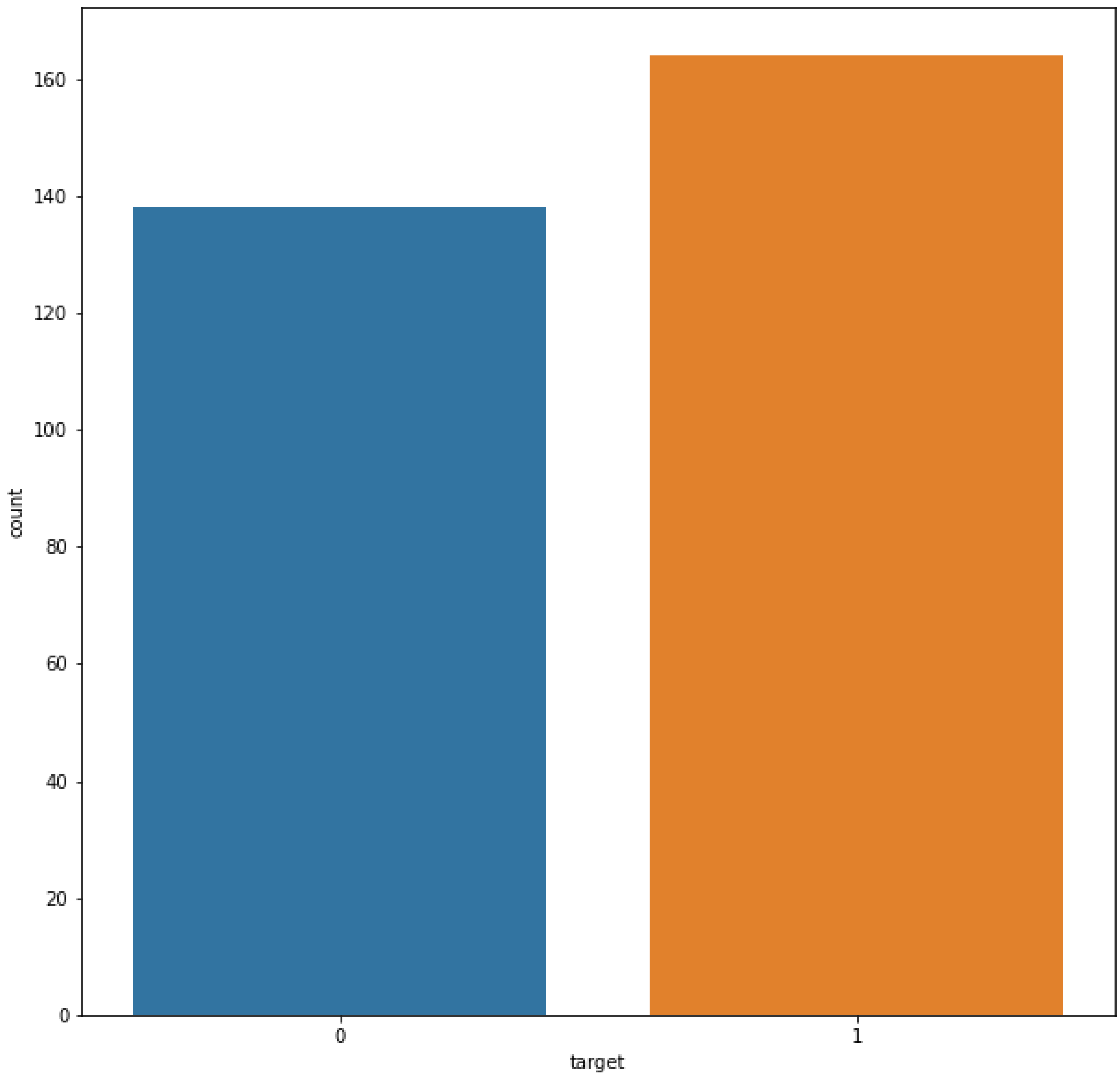
```
corr=heartData.corr()
corr.style.background_gradient(cmap='coolwarm')
```

```
<pandas.io.formats.style.Styler at 0x1d5bf13d100>
```

EXPLORATORY DATA ANALYSIS

```
y = heartData["target"]

sns.countplot(y)


target_temp = heartData.target.value_counts()

print(target_temp)
```

```
1    164
0    138
Name: target, dtype: int64
```



```
print("Percentage of patience without heart problems: 
"+str(round(target_temp[0]*100/1025,2)))
print("Percentage of patience with heart problems: 
"+str(round(target_temp[1]*100/1025,2)))

Percentage of patience without heart problems: 13.46
Percentage of patience with heart problems: 16.0
```

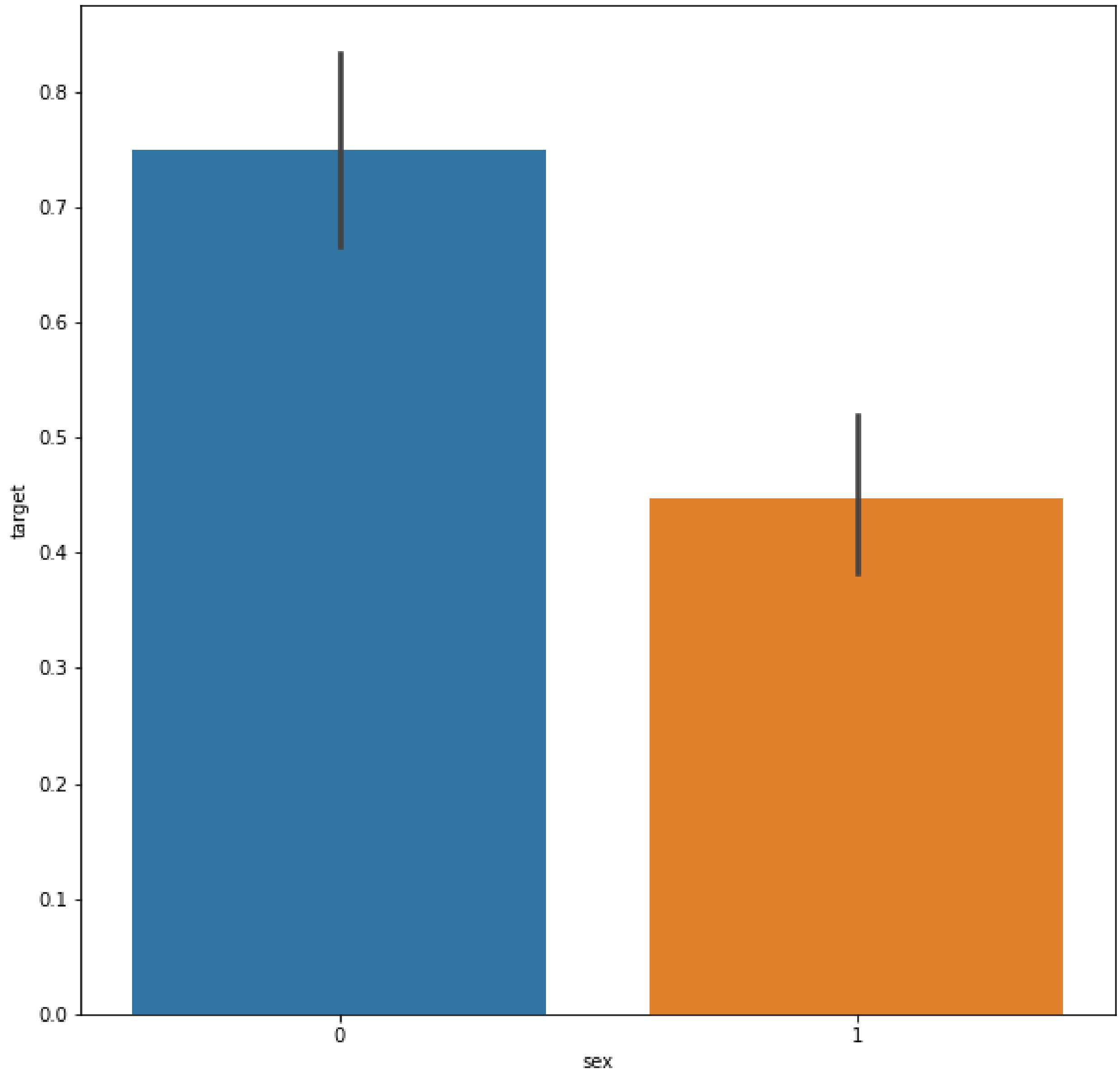
Analysing the 'Sex' feature

```
heartData["sex"].unique()
```

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

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

```
<AxesSubplot:xlabel='sex', ylabel='target'>
```



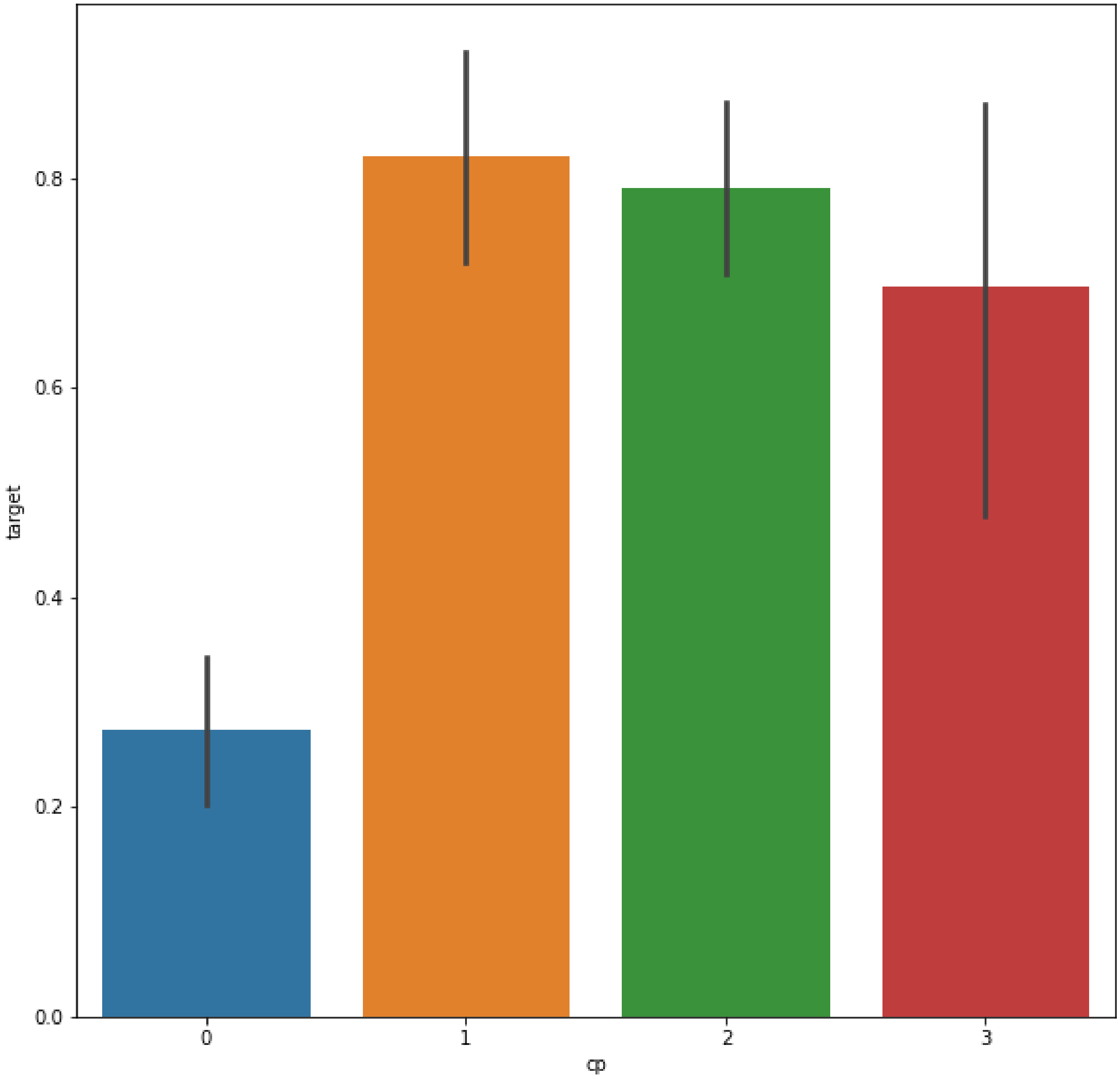
Analysing the 'Chest Pain Type' feature

```
heartData["cp"].unique()
```

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

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

```
<AxesSubplot:xlabel='cp', ylabel='target'>
```



Analysing the FBS feature

```
heartData["fbs"].describe()
```

```
count    302.000000
mean       0.149007
std        0.356686
min        0.000000
```

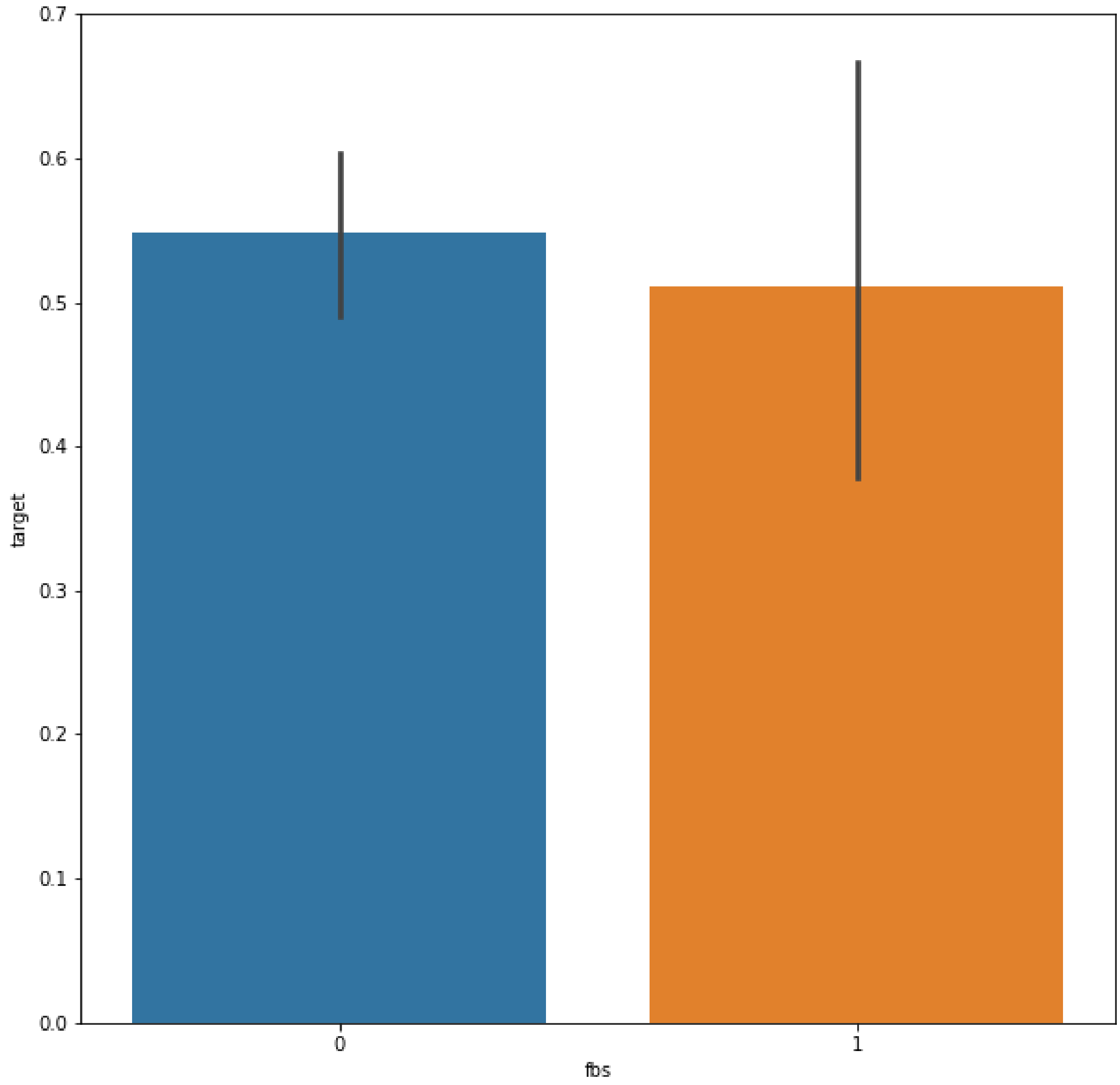
```
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000
Name: fbs, dtype: float64
```

```
heartData["fbs"].unique()
```

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

```
sns.barplot(heartData["fbs"],y)
```

```
<AxesSubplot:xlabel='fbs', ylabel='target'>
```



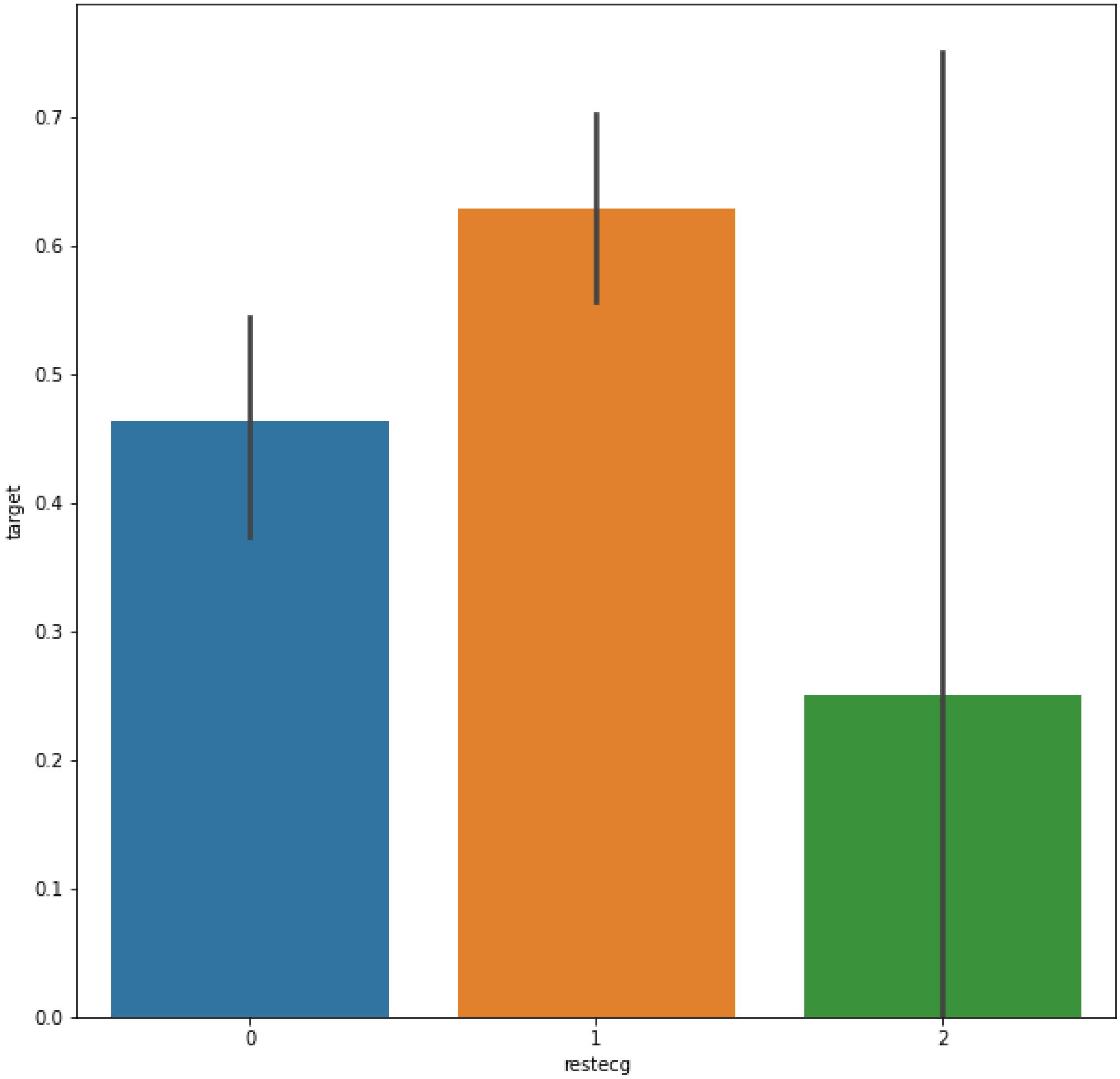
Analysing the restecg feature

```
heartData["restecg"].unique()
```

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

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

```
<AxesSubplot:xlabel='restecg', ylabel='target'>
```



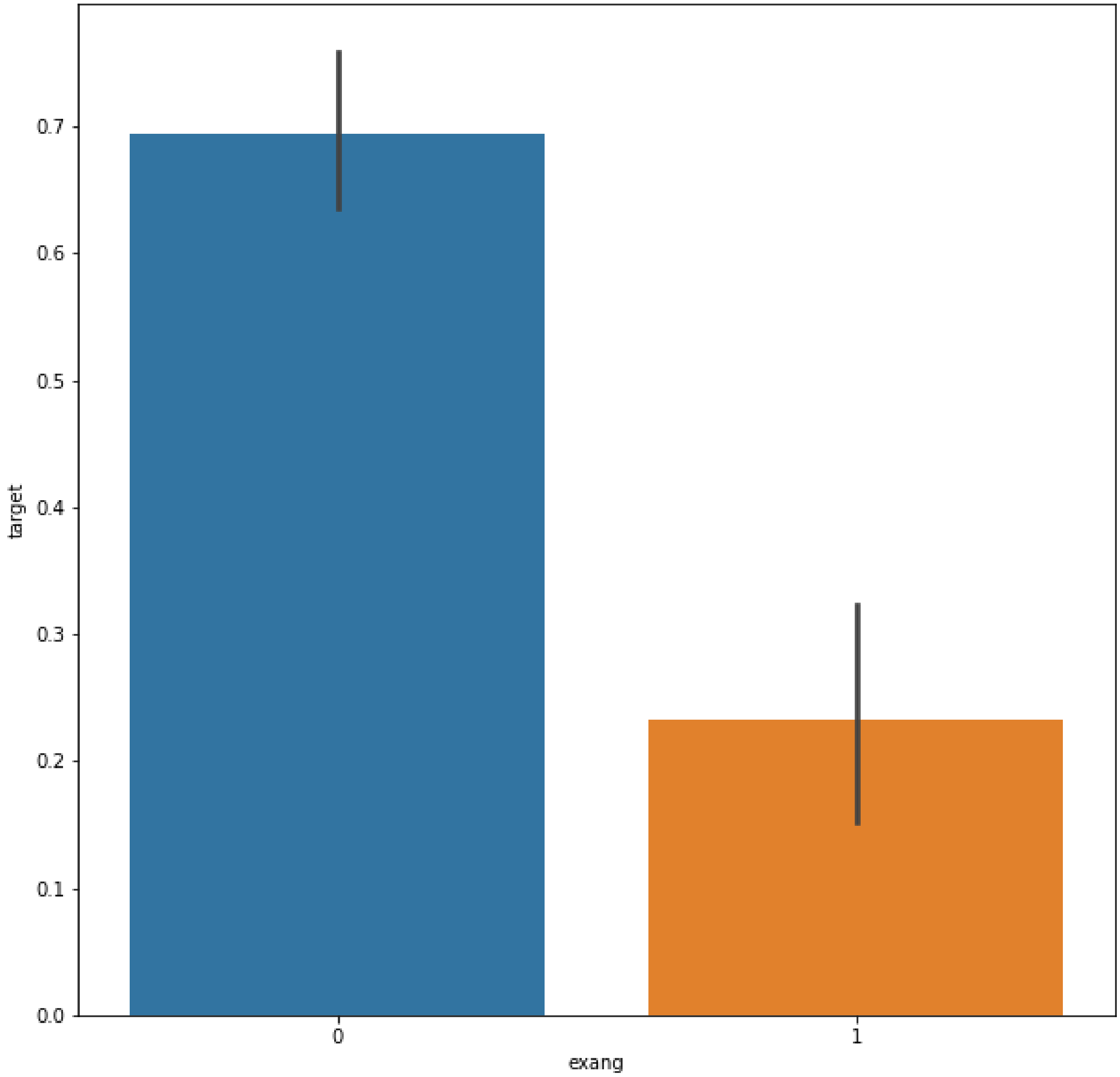
Analysing the 'exang' feature

```
heartData["exang"].unique()
```

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

```
sns.barplot(heartData["exang"],y)
```

```
<AxesSubplot:xlabel='exang', ylabel='target'>
```



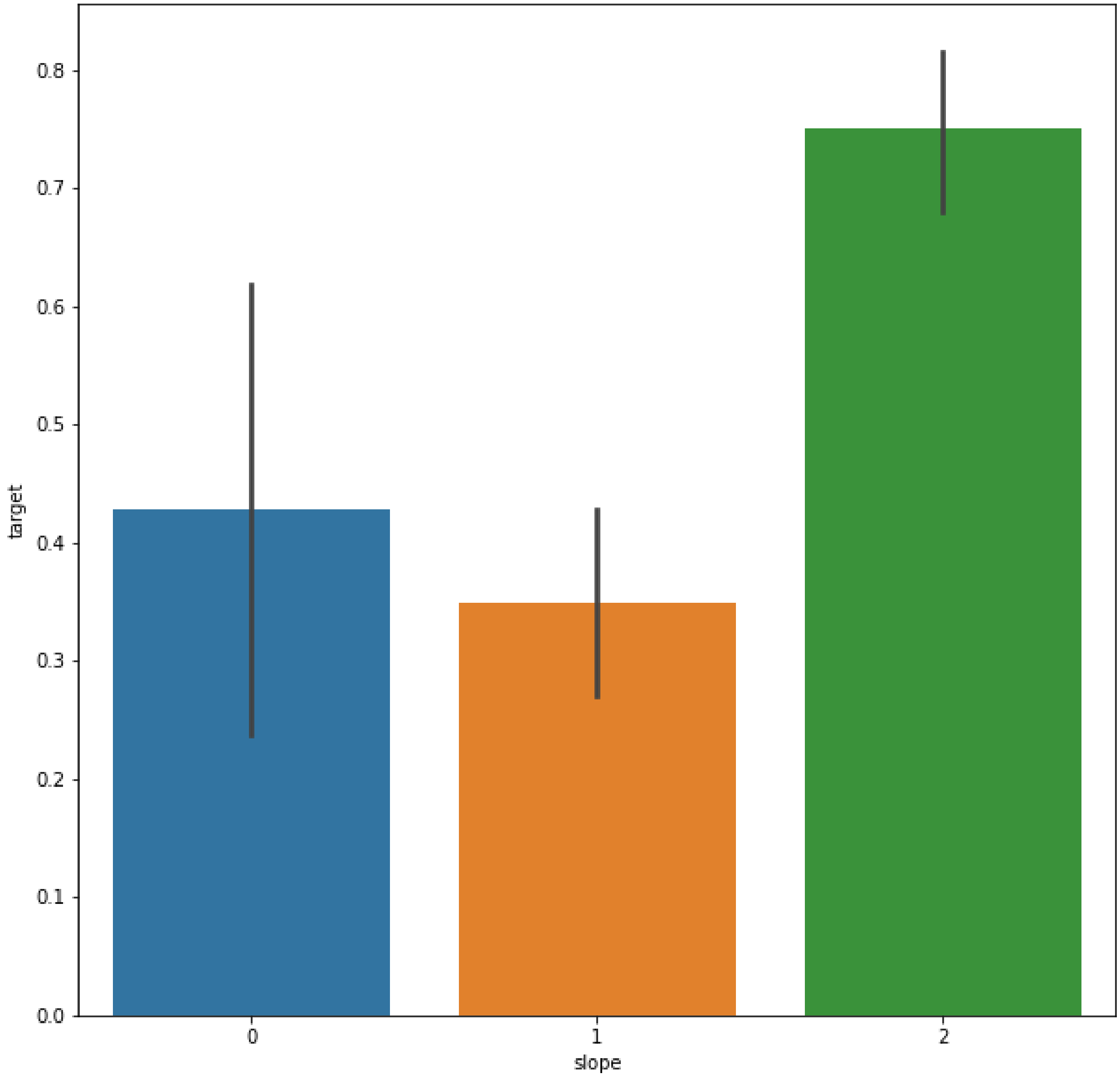
Analysing the 'Slope' feature

```
heartData["slope"].unique()
```

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

```
sns.barplot(heartData["slope"],y)
```

```
<AxesSubplot:xlabel='slope', ylabel='target'>
```



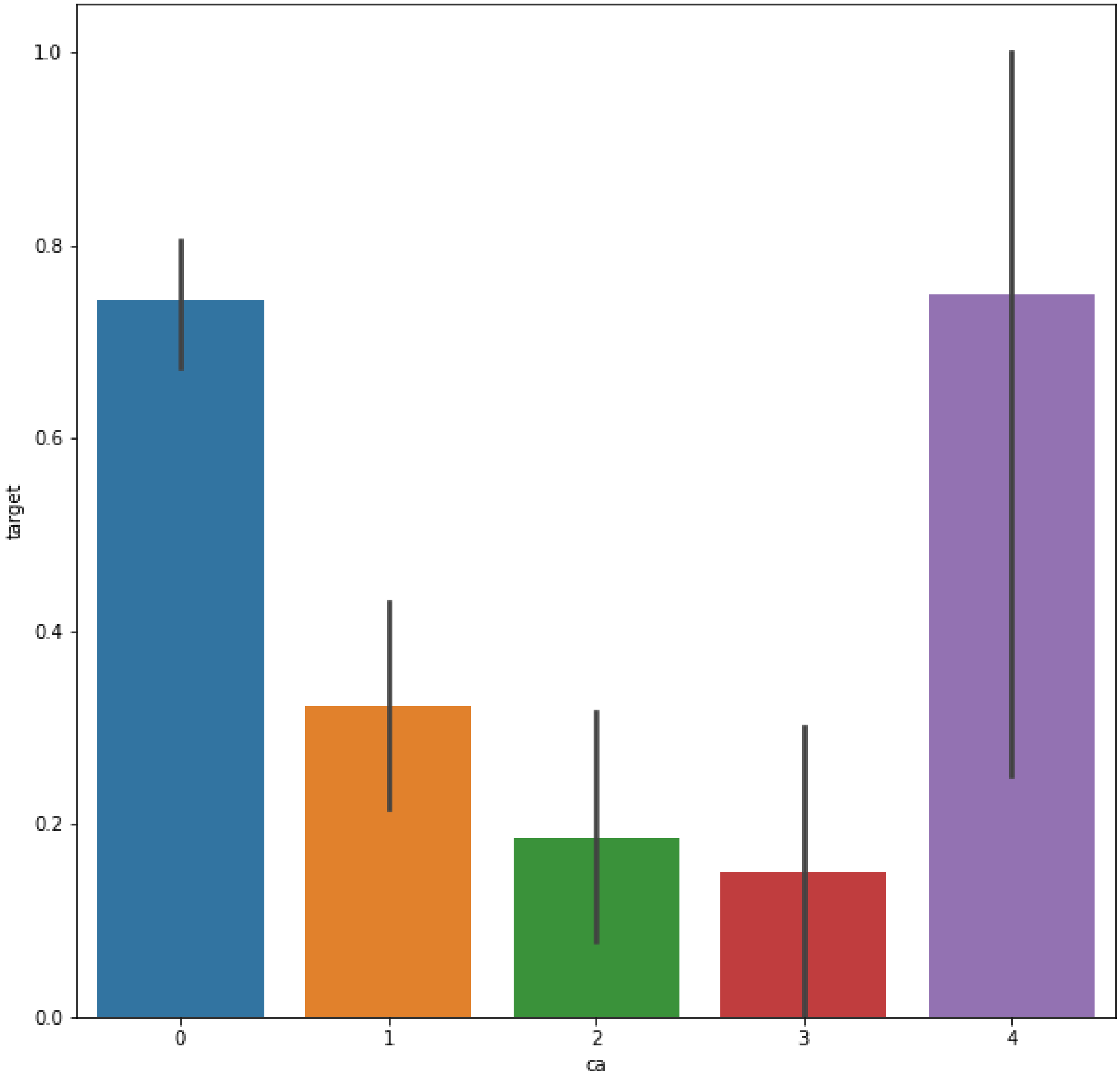
Analysing the 'ca' feature

```
heartData["ca"].unique()
```

```
array([2, 0, 1, 3, 4], dtype=int64)
```

```
sns.barplot(heartData["ca"],y)
```

```
<AxesSubplot:xlabel='ca', ylabel='target'>
```



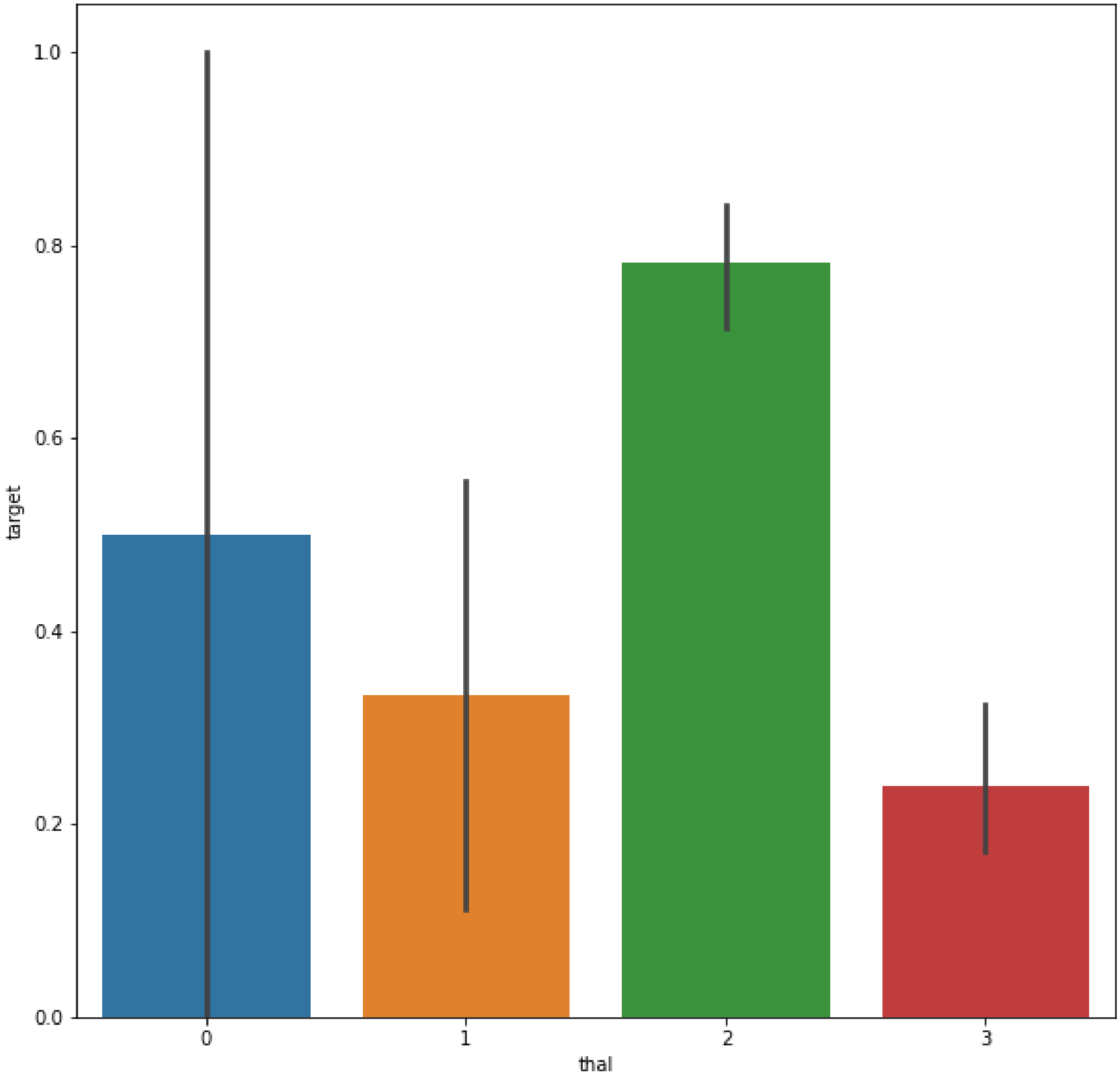
Analysing the 'thal' feature

```
 heartData["thal"].unique()
```

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

```
sns.barplot(heartData["thal"],y)
```

```
<AxesSubplot:xlabel='thal', ylabel='target'>
```



Train Test Split

```
from sklearn.model_selection import train_test_split

predictors = heartData.drop("target",axis=1)
target = heartData["target"]

X_train,X_test,Y_train,Y_test =
train_test_split(predictors,target,test_size=0.20,random_state=0)

X_train.shape

(241, 13)

X_test.shape

(61, 13)

Y_train.shape

(241,)

Y_test.shape

(61,)
```

Model Fitting

```
from sklearn.metrics import accuracy_score
```

Logistic Regression

```
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

(61,)

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: 83.61 %
```

# Naive Bayes

```
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

(61,)

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: 80.33 %
```

# K Nearest Neighbors

```
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

(61,)

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: 65.57 %
```

Decision Tree

```
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)


print(Y_pred_dt.shape)
```

```
(61,)
```

```
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: 85.25 %
```

# Random Forest

```
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)
```

```
Y_pred_rf.shape
```

```
(61,)
```

```
score_rf = round(accuracy_score(Y_pred_rf,Y_test)*100,2)

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

```
The accuracy score achieved using Decision Tree is: 86.89 %
```

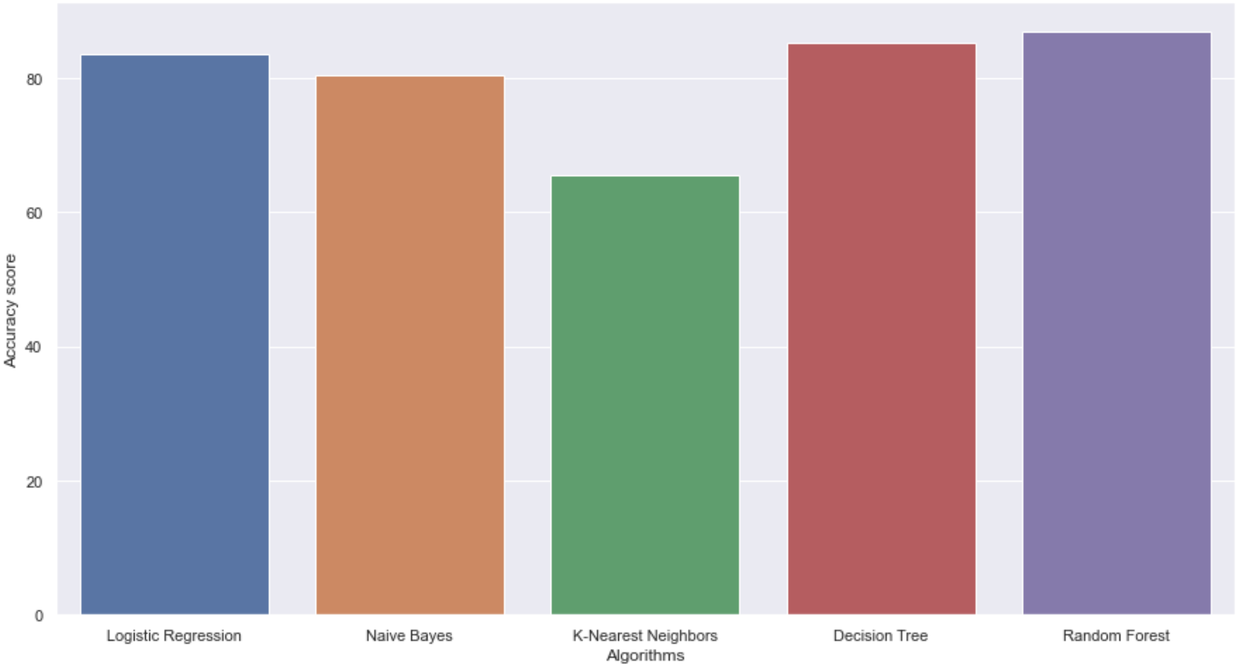
Output Final score

```
scores = [score_lr,score_nb,score_knn,score_dt,score_rf]
algorithms = ["Logistic Regression","Naive Bayes","K-Nearest 
Neighbors","Decision Tree","Random Forest"]

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: 83.61 %
The accuracy score achieved using Naive Bayes is: 80.33 %
The accuracy score achieved using K-Nearest Neighbors is: 65.57 %
The accuracy score achieved using Decision Tree is: 85.25 %
The accuracy score achieved using Random Forest is: 86.89 %
```

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

sns.barplot(algorithms,scores)
```

```
<AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>
```


0
20
40
60
80
Accuracy score
Logistic Regression
Naive Bayes
K-Nearest Neighbors
Decision Tree
Random Forest
Algorithms