In [2]:
```
import sklearn
import numpy as np
import pandas as pd
import plotly as plot
import plotly.express as px
import plotly.graph_objs as go

import cufflinks as cf
import matplotlib.pyplot as plt
import seaborn as sns
import os
from sklearn.metrics import accuracy_score
import plotly.offline as pyo
from plotly.offline import init_notebook_mode,plot,iplot
```

```
C:\Users\thero\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19: F
utureWarning:

pandas.util.testing is deprecated. Use the functions in the public API at pand
as.testing instead.
```

In [3]:
```
pyo.init_notebook_mode(connected=True)
cf.go_offline()
```

In [4]:
```
heart=pd.read_csv(heart.csv')
```

In [5]:
```
heart
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	ta
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

303 rows × 14 columns

In [6]:
```
info = ["age","1: male, 0: female","chest pain type, 1: typical angina, 2: at
```

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

```
age:                    age
sex:                    1: male, 0: female
cp:                     chest pain type, 1: typical angina, 2: atypical angin
a, 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 re
lative 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 def
ect
```

In [ ]:

In [7]:
```
heart['target']
```

Out[7]:
```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

In [8]:
```
heart.groupby('target').size()
```

Out[8]:
```
target
0    138
1    165
dtype: int64
```

In [9]:
```
heart.groupby('target').sum()
```

Out[9]:

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	7811	114	66	18547	34650	22	62	19196	76	218.8	161	161
1	8662	93	227	21335	39968	23	98	26147	23	96.2	263	60

In [10]:
```
heart.shape
```

Out[10]:
```
(303, 14)
```

In [11]:
```
heart.size
```

Out[11]:
```
4242
```

In [12]:
```
heart.describe()
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restec
count	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	0.52805
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.52586
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.00000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.00000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.00000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.00000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.00000

In [ ]:
```

```

In [13]:
```
heart.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
```

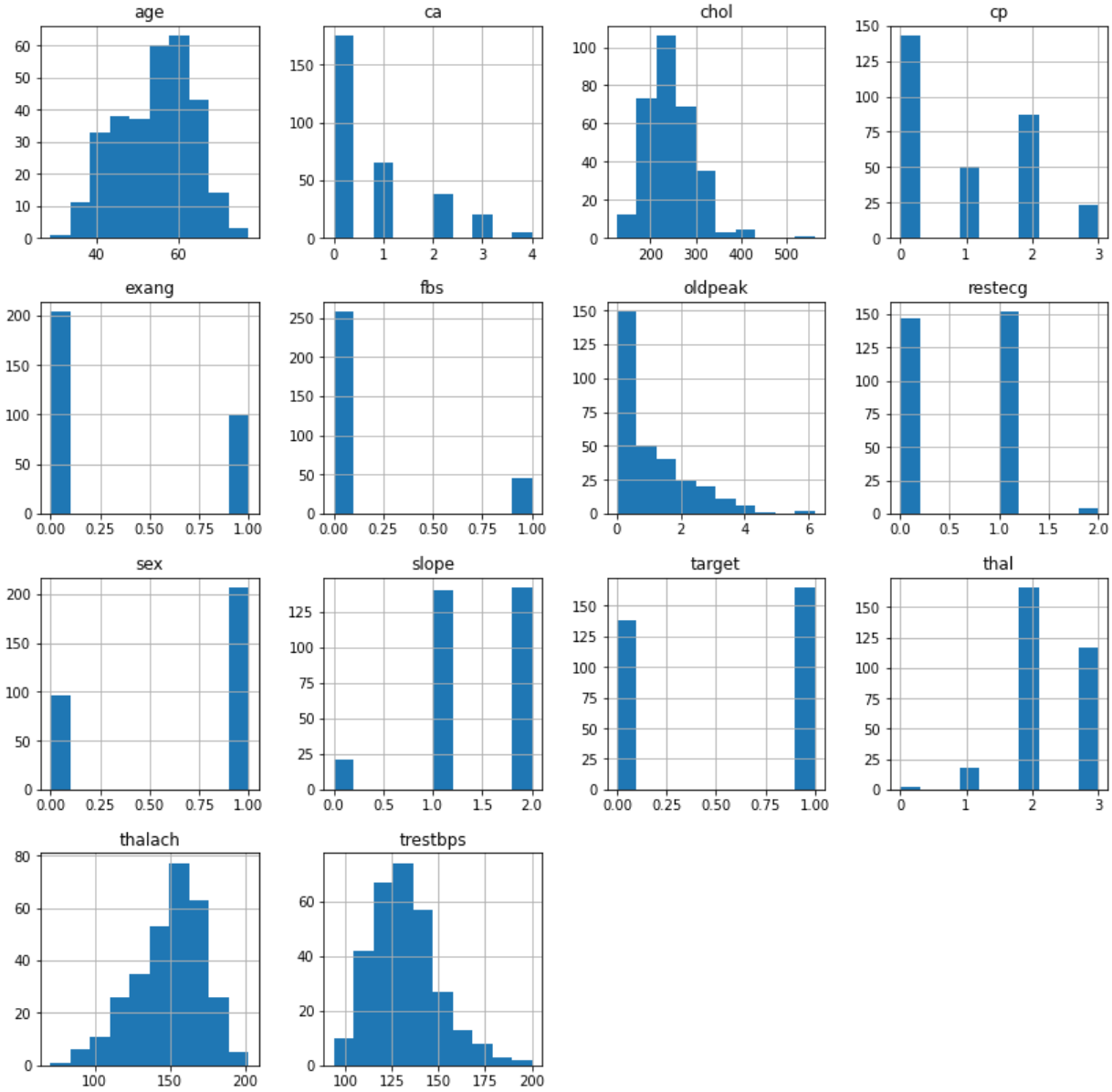
In [14]:
```
heart['target'].unique()
```

Out[14]:
```
array([1, 0], dtype=int64)
```

In [ ]:
```

```

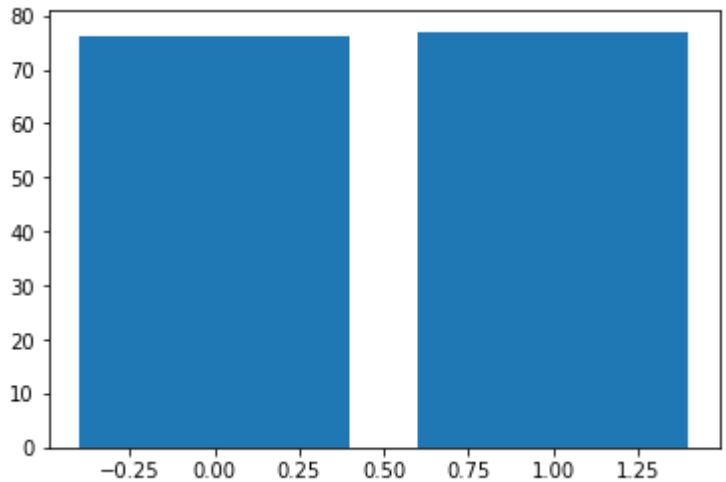
In [15]:
```
#Visualization
```

In [16]:
```
heart.hist(figsize=(14,14))
plt.show()
```

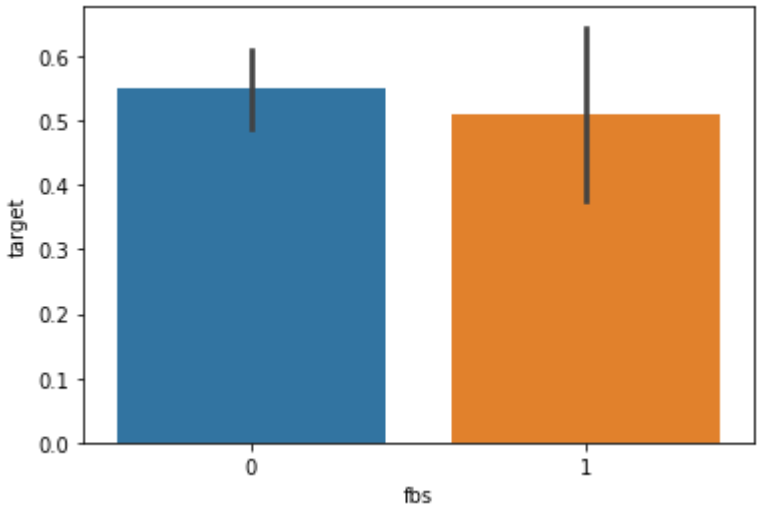


In [ ]:

In [17]:
```
plt.bar(x=heart['sex'],height=heart['age'])
plt.show()
```

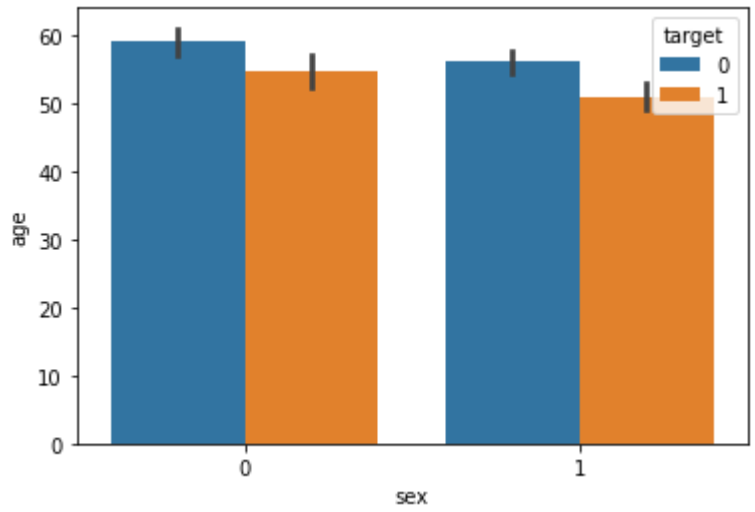


In [18]:
```
sns.barplot(x="fbs", y="target", data=heart)
plt.show()
```



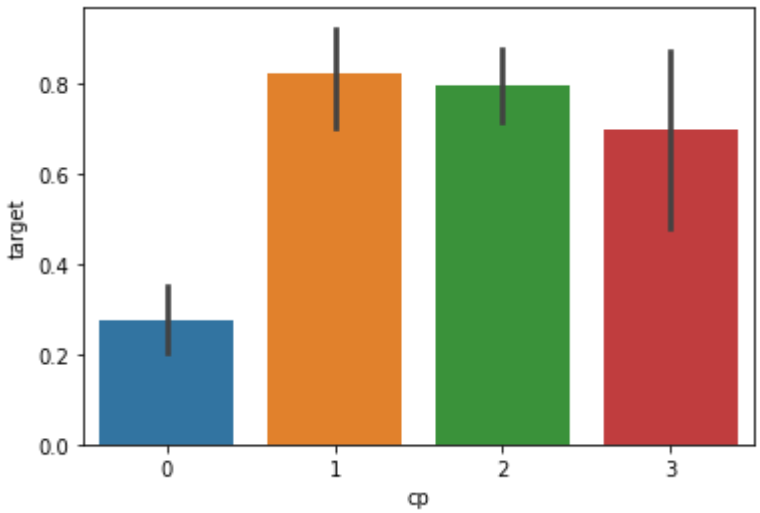
In [19]:
```
sns.barplot(x=heart['sex'],y=heart['age'],hue=heart['target'])
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x21093adb148>



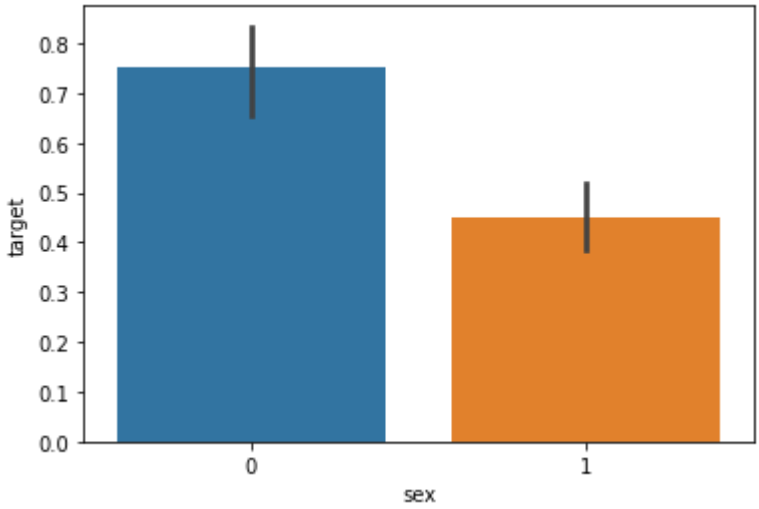
In [20]:
```
sns.barplot(heart["cp"],heart['target'])
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x2109351ca88>



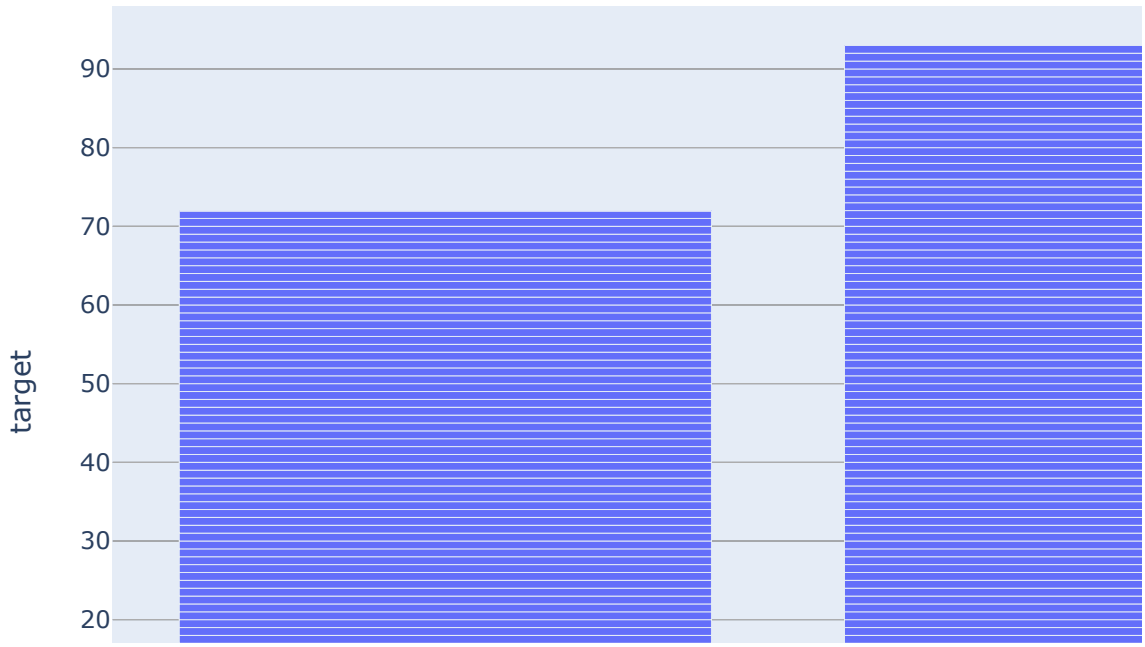
In [21]:
```
sns.barplot(heart["sex"],heart['target'])
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x21093b39308>



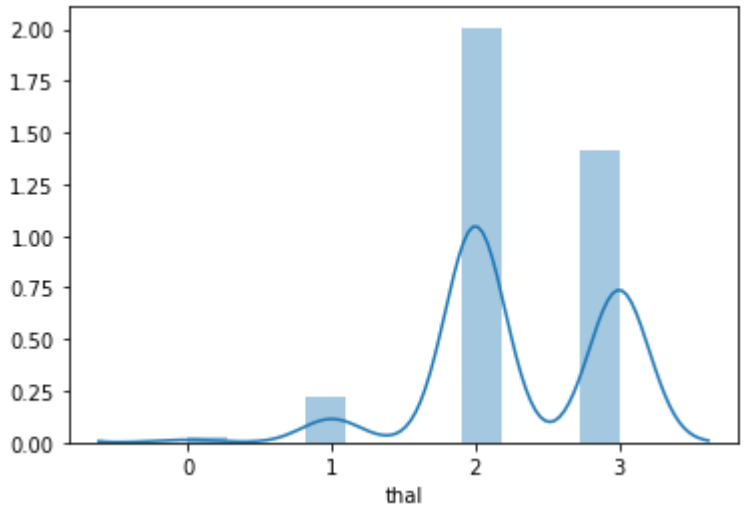
In [ ]:

In [25]:
```
px.bar(heart,heart['sex'],heart['target'])
```



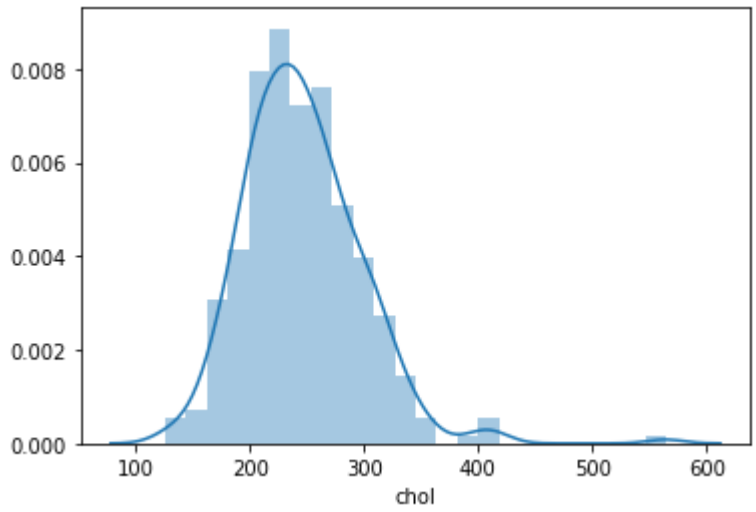
In [26]:
```
sns.distplot(heart["thal"])
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x21096778dc8>



In [27]:
```
sns.distplot(heart["chol"])
```

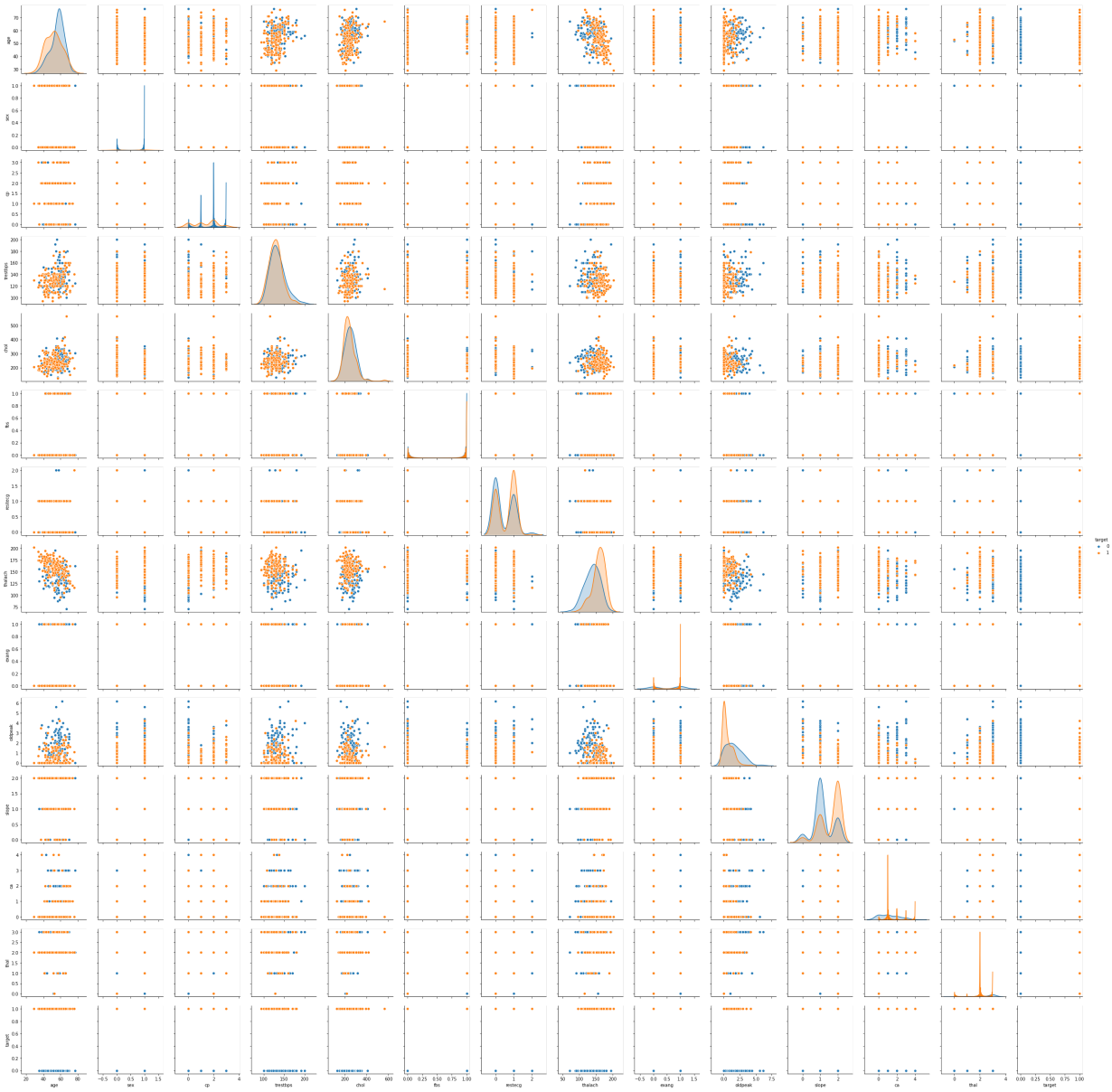
Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21096778108>
```



In [28]:
```
sns.pairplot(heart,hue='target')
```

```
C:\Users\thero\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:48
7: RuntimeWarning:

invalid value encountered in true_divide

C:\Users\thero\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.
py:34: RuntimeWarning:

invalid value encountered in double_scalars

C:\Users\thero\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:48
7: RuntimeWarning:

invalid value encountered in true_divide

C:\Users\thero\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.
py:34: RuntimeWarning:

invalid value encountered in double_scalars
```

Out[28]:
```
<seaborn.axisgrid.PairGrid at 0x21096a2d0c8>
```



In [29]:
```
heart
```

Out[29]:

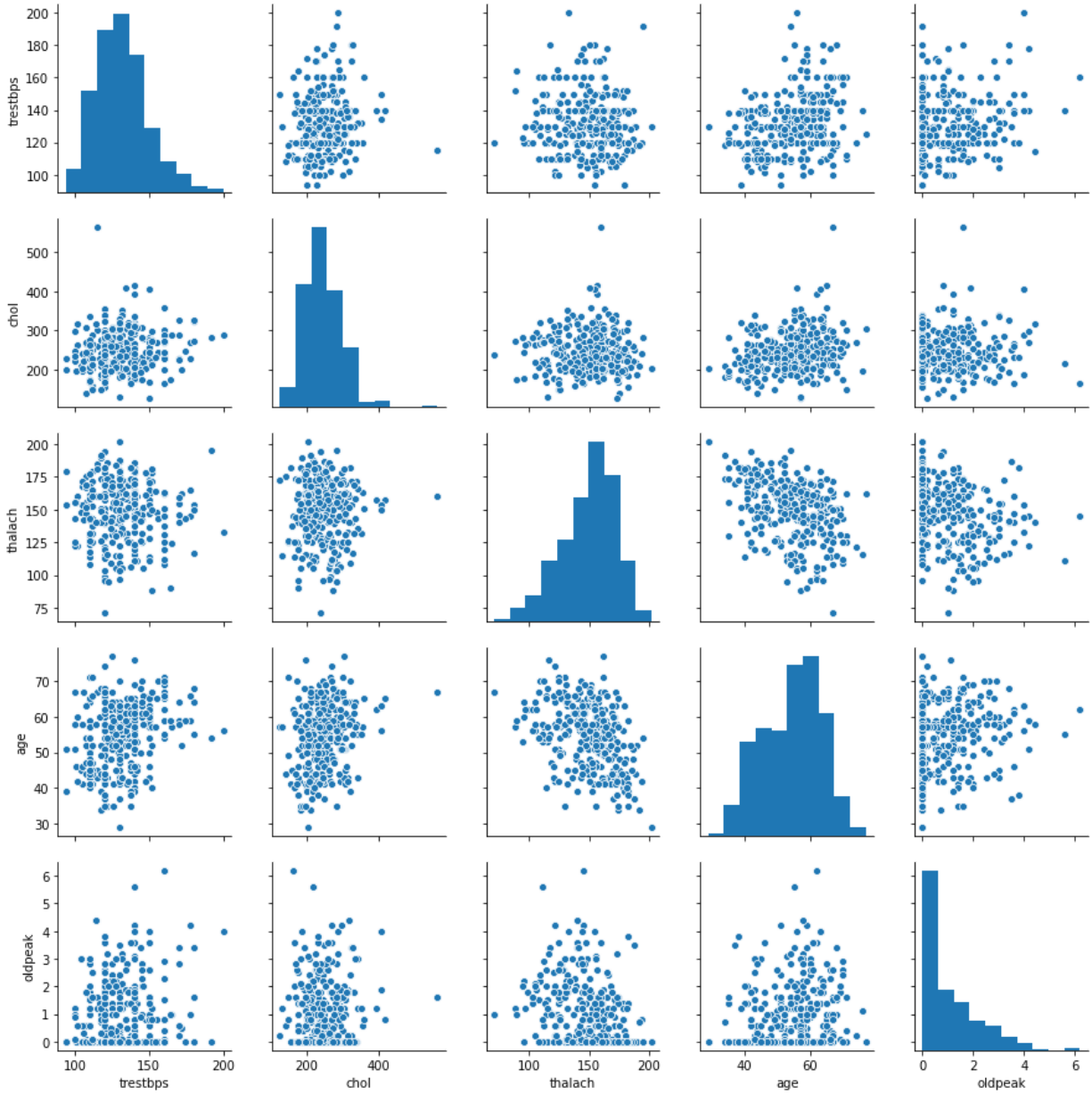
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	ta
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

303 rows × 14 columns

In [30]:
```
numeric_columns=['trestbps','chol','thalach','age','oldpeak']
```

In [31]:
```
sns.pairplot(heart[numeric_columns])
```

Out[31]:
```
<seaborn.axisgrid.PairGrid at 0x21096ca2188>
```



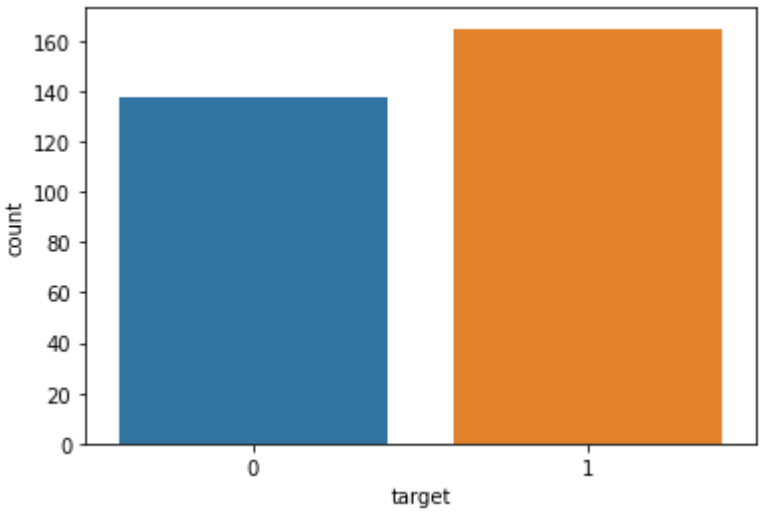
In [32]:
```
heart['target']
```

Out[32]:
```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

In [33]:
```
y = heart["target"]
```

```
sns.countplot(y)

target_temp = heart.target.value_counts()

print(target_temp)
```

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



In [ ]:

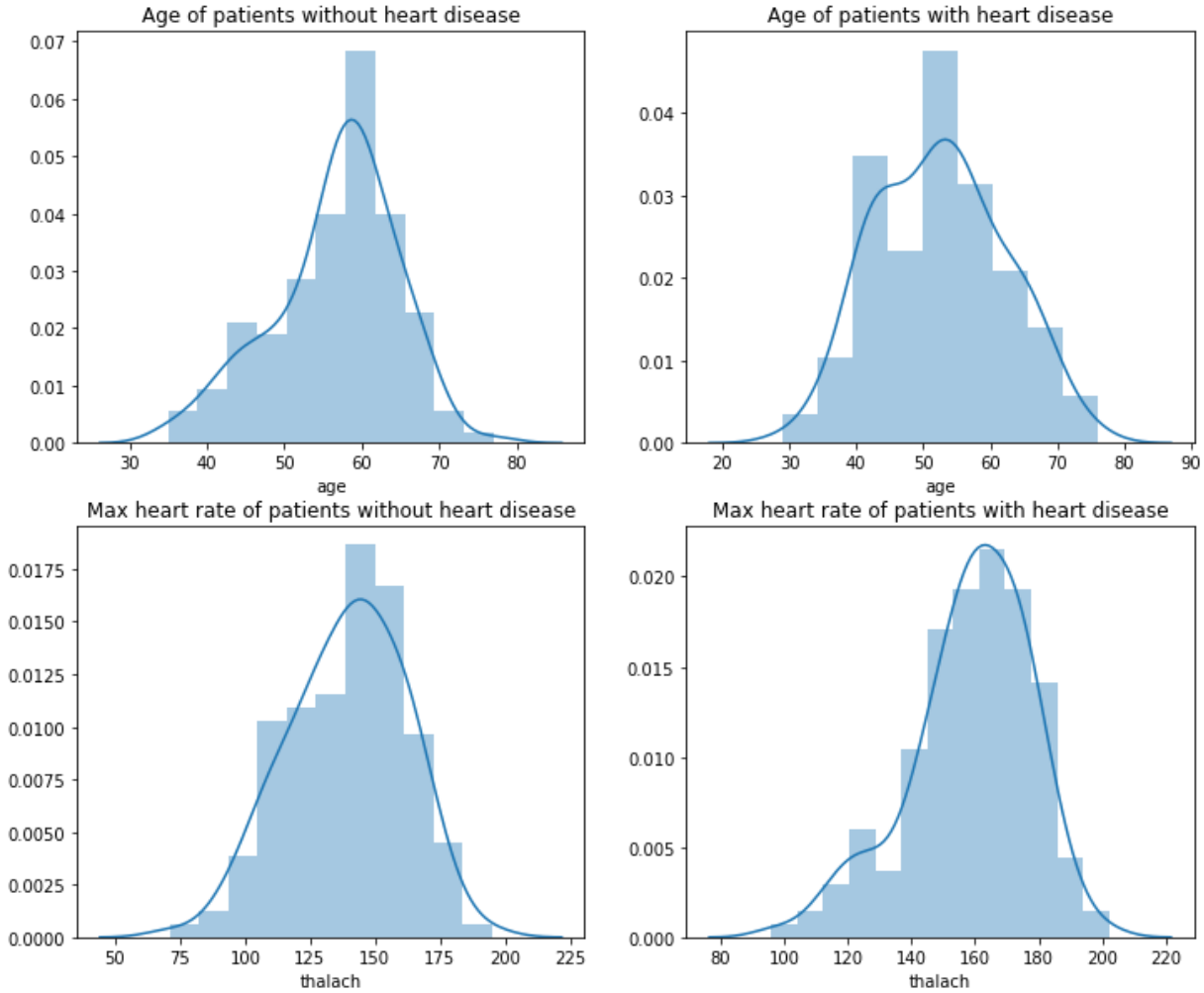
In [34]:

```
# create a correlation heatmap
sns.heatmap(heart[numeric_columns].corr(),annot=True, cmap='terrain', linewid
fig=plt.gcf()
fig.set_size_inches(8,6)
plt.show()
```

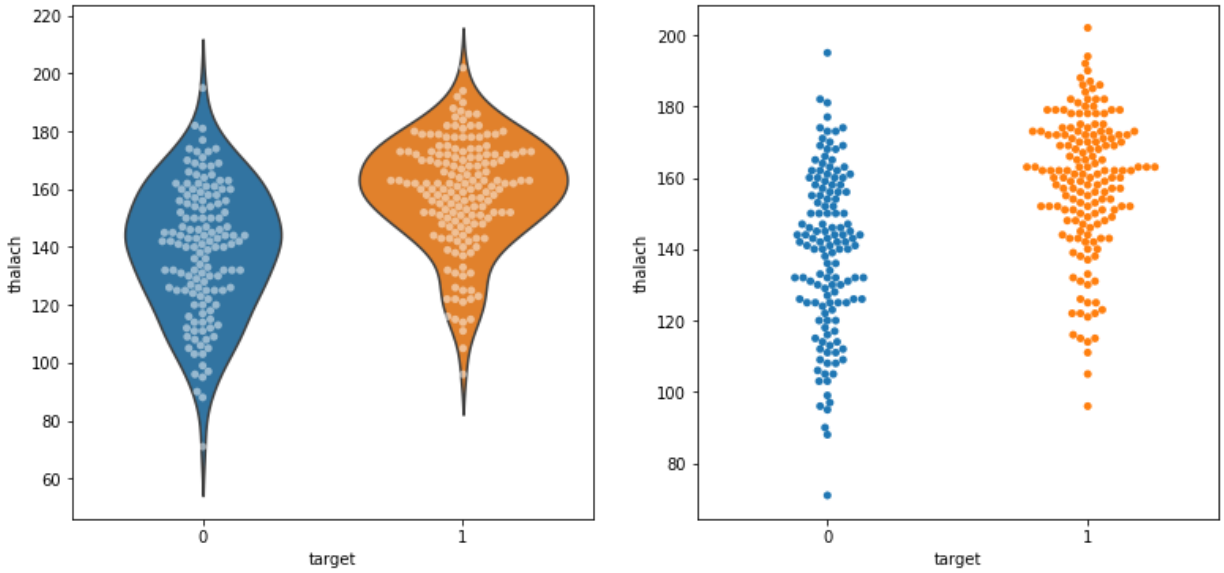
In [ ]:

In [ ]:

In [35]:
```
# create four distplots
plt.figure(figsize=(12,10))
plt.subplot(221)
sns.distplot(heart[heart['target']==0].age)
plt.title('Age of patients without heart disease')
plt.subplot(222)
sns.distplot(heart[heart['target']==1].age)
plt.title('Age of patients with heart disease')
plt.subplot(223)
sns.distplot(heart[heart['target']==0].thalach )
plt.title('Max heart rate of patients without heart disease')
plt.subplot(224)
sns.distplot(heart[heart['target']==1].thalach )
plt.title('Max heart rate of patients with heart disease')
plt.show()
```



In [36]:
```
plt.figure(figsize=(13,6))
plt.subplot(121)
sns.violinplot(x="target", y="thalach", data=heart, inner=None)
sns.swarmplot(x="target", y="thalach", data=heart, color='w', alpha=0.5)


plt.subplot(122)
sns.swarmplot(x="target", y="thalach", data=heart)
plt.show()
```



In [ ]:

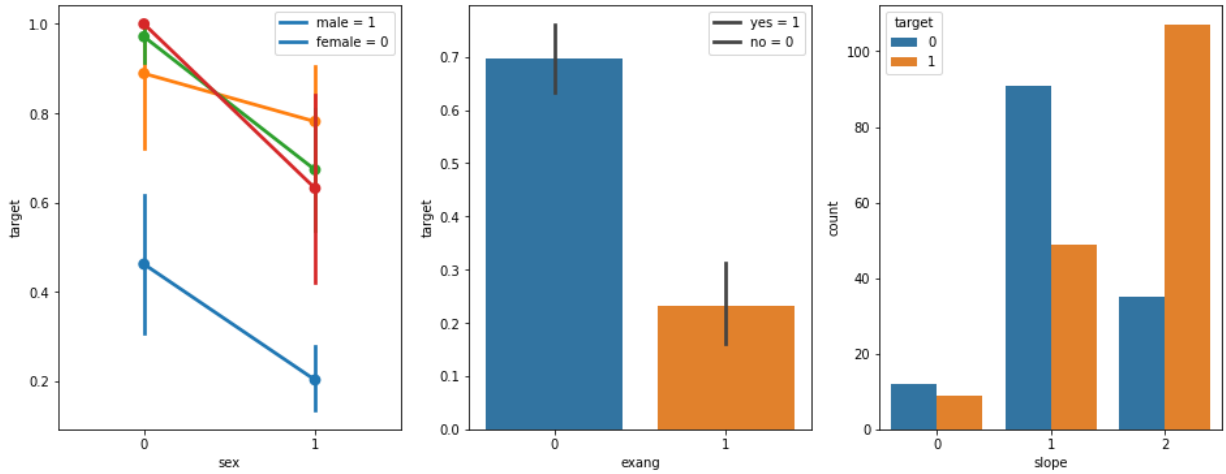
In [37]:
```
heart
```

Out[37]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	ta
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

303 rows × 14 columns

In [38]:
```
# create pairplot and two barplots
plt.figure(figsize=(16,6))
plt.subplot(131)
sns.pointplot(x="sex", y="target", hue='cp', data=heart)
plt.legend(['male = 1', 'female = 0'])
plt.subplot(132)
sns.barplot(x="exang", y="target", data=heart)
plt.legend(['yes = 1', 'no = 0'])
plt.subplot(133)
sns.countplot(x="slope", hue='target', data=heart)
plt.show()
```



In [ ]:

In [ ]:

In [39]:
```
#DATA Preprocessing
```

In [40]:
```
##############################################################################
```

In [41]:
```
heart['target'].value_counts()
```

Out[41]:
```
1    165
0    138
Name: target, dtype: int64
```

In [42]:
```
heart['target'].isnull()
```

Out[42]:
```
0      False
1      False
2      False
3      False
4      False
       ...
298    False
299    False
300    False
301    False
302    False
Name: target, Length: 303, dtype: bool
```

In [43]:
```
heart['target'].sum()
```

Out[43]:
```
165
```

In [44]:
```
heart['target'].unique()
```

Out[44]:
```
array([1, 0], dtype=int64)
```

In [45]:
```
##############################################################################
```

In [ ]:

In [46]:
```
heart.isnull().sum()
```

Out[46]:
```
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 [ ]:

In [ ]:

In [47]:
```
#Storing in X and y
```

In [48]:
```
X,y=heart.loc[:,:'thal'],heart.loc[:,'target']
```

In [49]:
```
X
```

Out[49]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

In [50]:
```
y
```

Out[50]:
```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

In [51]:
```
####Or X, y = heart.iloc[:, :-1], heart.iloc[:, -1]
```

In [52]:
```
X.shape
```

Out[52]:
```
(303, 13)
```

In [53]:
```
y.shape
```

Out[53]:
```
(303,)
```

In [54]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [55]:
```
X=heart.drop(['target'],axis=1)
```

In [56]:
```
#X=np.array(X)
```

In [57]:
```
X
```

Out[57]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

In [58]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=10,test_size=
```

In [59]:
```
X_test
```

Out[59]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
246	56	0	0	134	409	0	0	150	1	1.9	1	2	3
183	58	1	2	112	230	0	0	165	0	2.5	1	1	3
229	64	1	2	125	309	0	1	131	1	1.8	1	0	3
126	47	1	0	112	204	0	1	143	0	0.1	2	0	2
184	50	1	0	150	243	0	0	128	0	2.6	1	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
69	62	0	0	124	209	0	1	163	0	0.0	2	0	2
21	44	1	2	130	233	0	1	179	1	0.4	2	0	2
210	57	1	2	128	229	0	0	150	0	0.4	1	1	3
78	52	1	1	128	205	1	1	184	0	0.0	2	0	2
174	60	1	0	130	206	0	0	132	1	2.4	1	2	3

91 rows × 13 columns

In [60]:
```
y_test
```

Out[60]:
```
246    0
183    0
229    0
126    1
184    0
      ..
69     1
21     1
210    0
78     1
174    0
Name: target, Length: 91, dtype: int64
```

In [ ]:
```

```

In [ ]:
```

```

In [61]:
```
print ("train_set_x shape: " + str(X_train.shape))
print ("train_set_y shape: " + str(y_train.shape))
print ("test_set_x shape: " + str(X_test.shape))
print ("test_set_y shape: " + str(y_test.shape))
```

```
train_set_x shape: (212, 13)
train_set_y shape: (212,)
test_set_x shape: (91, 13)
test_set_y shape: (91,)
```

In [ ]:
```

```

In [ ]:

In [ ]:

In [62]:
```
#Model
```

In [63]:
```
#Decision Tree Classifier
```

In [64]:
```
Catagory=['No....but i pray you get Heart Disease or at leaset Corona Virus S
```

In [65]:
```
from sklearn.tree import DecisionTreeClassifier


dt=DecisionTreeClassifier()
dt.fit(X_train,y_train)
```

Out[65]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [66]:
```
prediction=dt.predict(X_test)
accuracy_dt=accuracy_score(y_test,prediction)*100
```

In [67]:
```
accuracy_dt
```

Out[67]:
```
79.12087912087912
```

In [68]:
```
print("Accuracy on training set: {:.3f}".format(dt.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(dt.score(X_test, y_test)))
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.791
```

In [ ]:

In [69]:
```
y_test
```

Out[69]:
```
246    0
183    0
229    0
126    1
184    0
      ..
69     1
21     1
210    0
78     1
174    0
Name: target, Length: 91, dtype: int64
```

In [70]:
```
prediction
```

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

In [ ]:
```

```

In [71]:
```
X_DT=np.array([[63 ,1, 3,145,233,1,0,150,0,2.3,0,0,1]])
X_DT_prediction=dt.predict(X_DT)
```

In [72]:
```
X_DT_prediction[0]
```

Out[72]:
```
1
```

In [73]:
```
print(Catagory[int(X_DT_prediction[0])])
```

```
Yes you have Heart Disease....RIP in Advance
```

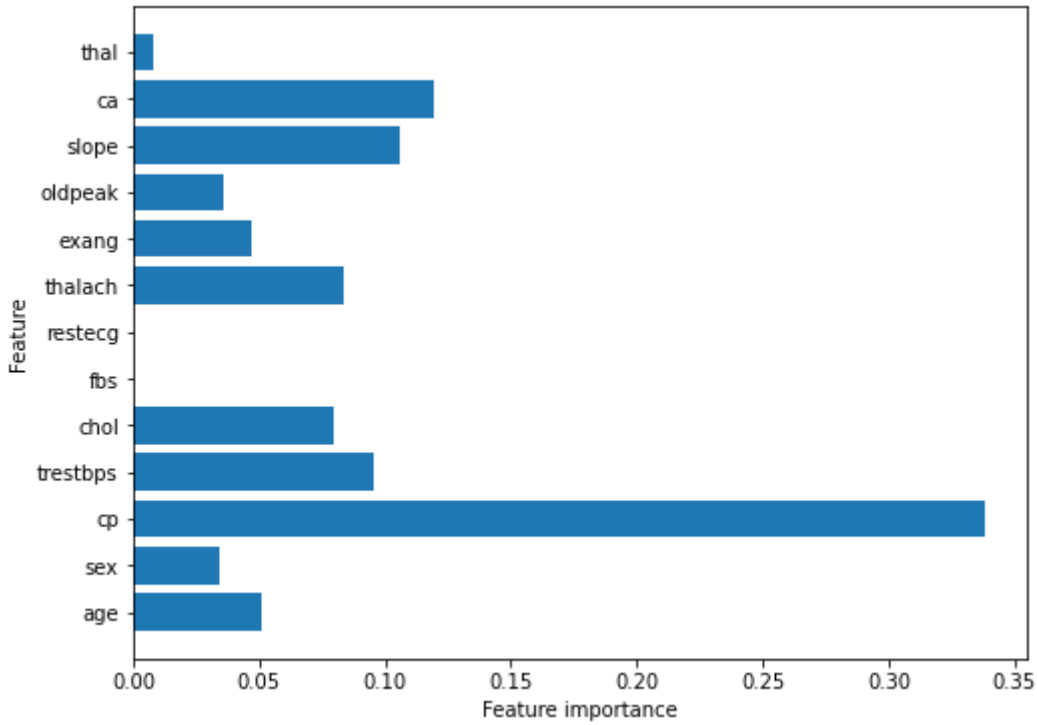
In [ ]:
```

```

In [74]:
```
#Feature Importance in Decision Trees
```

In [75]:
```
print("Feature importances:\n{}".format(dt.feature_importances_))
```

```
Feature importances:
[0.05112253 0.03461456 0.33832546 0.09527361 0.0799599  0.
 0.         0.08395957 0.04724994 0.0362058  0.10560028 0.11946876
 0.0082196 ]
```

In [76]:
```
def plot_feature_importances_diabetes(model):
    plt.figure(figsize=(8,6))
    n_features = 13
    plt.barh(range(n_features), model.feature_importances_, align='center')
    plt.yticks(np.arange(n_features), X)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")
    plt.ylim(-1, n_features)
plot_feature_importances_diabetes(dt)
plt.savefig('feature_importance')
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [77]:
```
# KNN
```

In [78]:
```
sc=StandardScaler().fit(X_train)
X_train_std=sc.transform(X_train)
X_test_std=sc.transform(X_test)
```

In [79]:
```
X_test_std
```

Out[79]:
```
array([[ 0.18111199, -1.35154233, -0.97043553, ..., -0.6067969 ,
         1.33369489,  1.22676132],
       [ 0.39865161,  0.73989544,  0.97963397, ..., -0.6067969 ,
         0.33105902,  1.22676132],
       [ 1.05127045,  0.73989544,  0.97963397, ..., -0.6067969 ,
        -0.67157686,  1.22676132],
```

```
       ...,
       [ 0.2898818 ,  0.73989544,  0.97963397, ..., -0.6067969 ,
         0.33105902,  1.22676132],
       [-0.25396724,  0.73989544,  0.00459922, ...,  0.98136289,
        -0.67157686, -0.41927286],
       [ 0.61619122,  0.73989544, -0.97043553, ..., -0.6067969 ,
         1.33369489,  1.22676132]])
```

In [80]:
```
from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train_std,y_train)
```

Out[80]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                     weights='uniform')
```

In [81]:
```
prediction_knn=knn.predict(X_test_std)
accuracy_knn=accuracy_score(y_test,prediction_knn)*100
```

In [82]:
```
accuracy_knn
```

Out[82]:
```
84.61538461538461
```

In [83]:
```
print("Accuracy on training set: {:.3f}".format(knn.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(knn.score(X_test, y_test)))
```

```
Accuracy on training set: 0.373
Accuracy on test set: 0.516
```

In [ ]:
```

```

In [84]:
```
k_range=range(1,26)
scores={}
scores_list=[]

for k in k_range:
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_std,y_train)
    prediction_knn=knn.predict(X_test_std)
    scores[k]=accuracy_score(y_test,prediction_knn)
    scores_list.append(accuracy_score(y_test,prediction_knn))
```

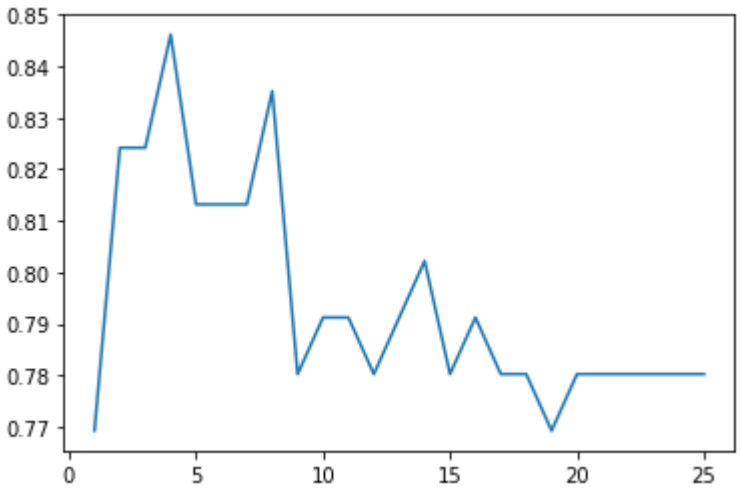
In [85]:
```
scores
```

Out[85]:
```
{1: 0.7692307692307693,
 2: 0.8241758241758241,
 3: 0.8241758241758241,
 4: 0.8461538461538461,
 5: 0.8131868131868132,
 6: 0.8131868131868132,
 7: 0.8131868131868132,
 8: 0.8351648351648352,
 9: 0.7802197802197802,
 10: 0.7912087912087912,
 11: 0.7912087912087912,
 12: 0.7802197802197802,
 13: 0.7912087912087912,
```

```
14: 0.8021978021978022,
15: 0.7802197802197802,
16: 0.7912087912087912,
17: 0.7802197802197802,
18: 0.7802197802197802,
19: 0.7692307692307693,
20: 0.7802197802197802,
21: 0.7802197802197802,
22: 0.7802197802197802,
23: 0.7802197802197802,
24: 0.7802197802197802,
25: 0.7802197802197802}
```

In [86]:
```
plt.plot(k_range,scores_list)
```

Out[86]:
```
[<matplotlib.lines.Line2D at 0x210a2f2d1c8>]
```



In [87]:
```
px.line(x=k_range,y=scores_list)
```

In [ ]:

```
```

In [ ]:

```
```

In [88]:

```
X_knn=np.array([[63 ,1, 3,145,233,1,0,150,0,2.3,0,0,1]])
X_knn_std=sc.transform(X_knn)
X_knn_prediction=dt.predict(X_knn)
```

In [89]:

```
X_knn_std
```

Out[89]:

```
array([[ 0.94250064,  0.73989544,  1.95466871,  0.75961822, -0.30064937,
         2.37170825, -0.9841849 ,  0.01848325, -0.6723502 ,  1.10653103,
        -2.1949567 , -0.67157686, -2.06530703]])
```

In [90]:

```
(X_knn_prediction[0])
```

Out[90]:

```
1
```

In [91]:

```
print(Catagory[int(X_knn_prediction[0])])
```

```
Yes you have Heart Disease....RIP in Advance
```

In [ ]:

```
```

In [92]:

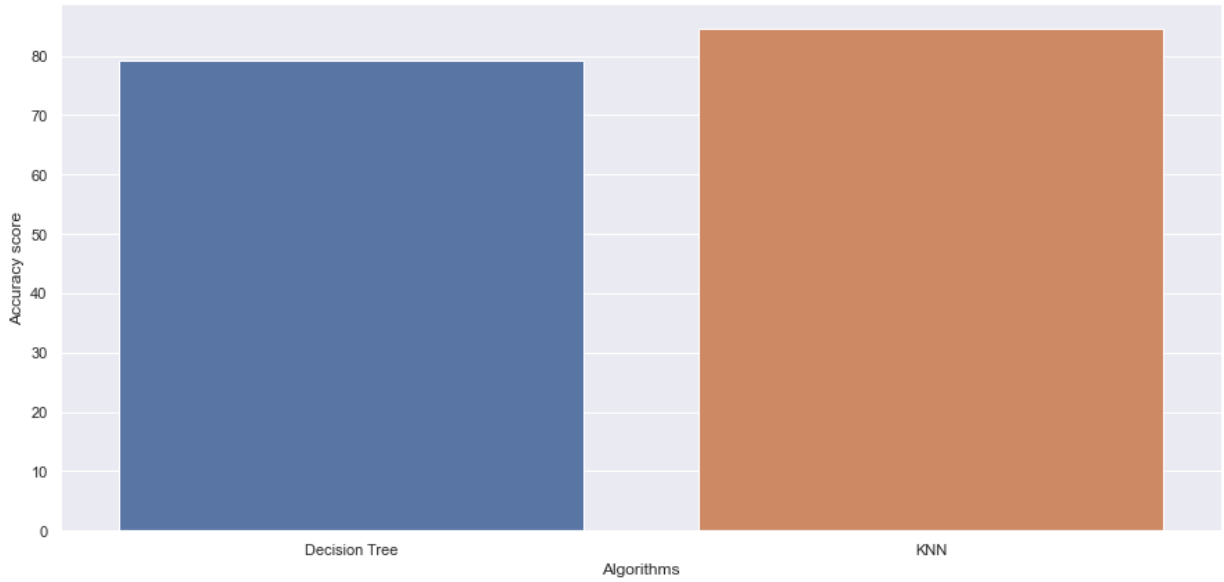
```
algorithms=['Decision Tree','KNN']
scores=[accuracy_dt,accuracy_knn]
```

In [93]:

```
sns.set(rc={'figure.figsize':(15,7)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

Out[93]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2109fb90188>
```


80
70
60
50
40
30
20
10
0
Accuracy score
Decision Tree
KNN
Algorithms

In [ ]:

In [ ]:

In [ ]: