In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import os
os.chdir(r"C:\Users\varshini rajkumar\Desktop")
df=pd.read_csv("heart.csv")
```

In [3]:
```
df.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
```

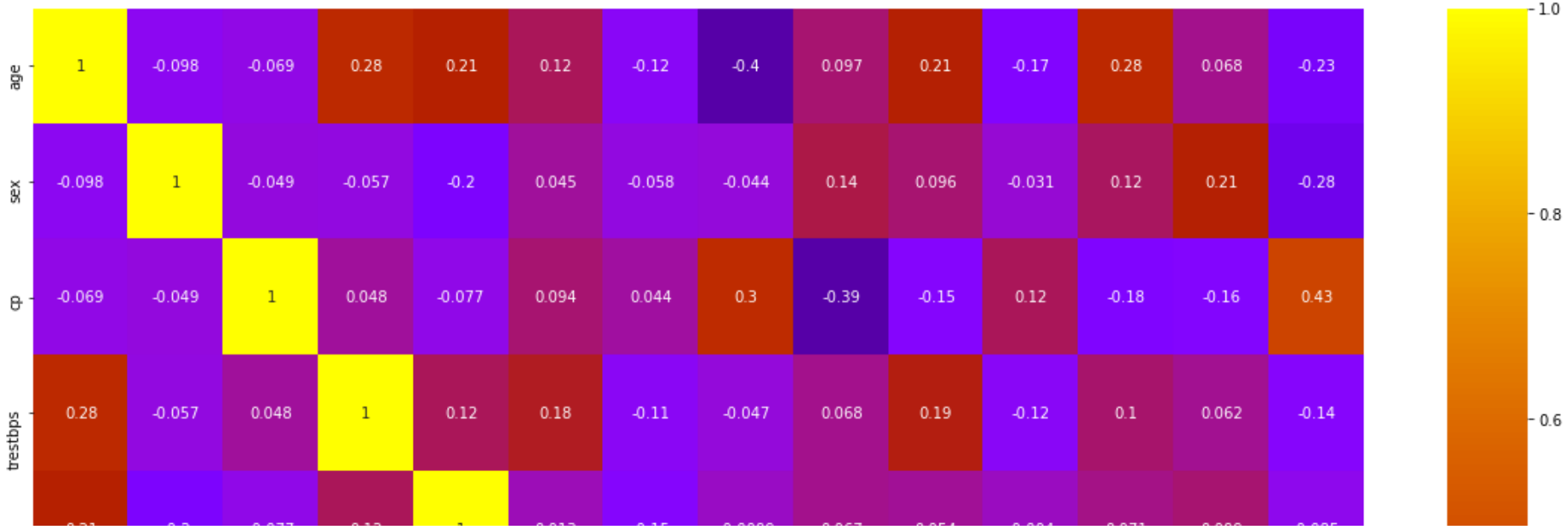
In [4]:
```
df.describe()
```

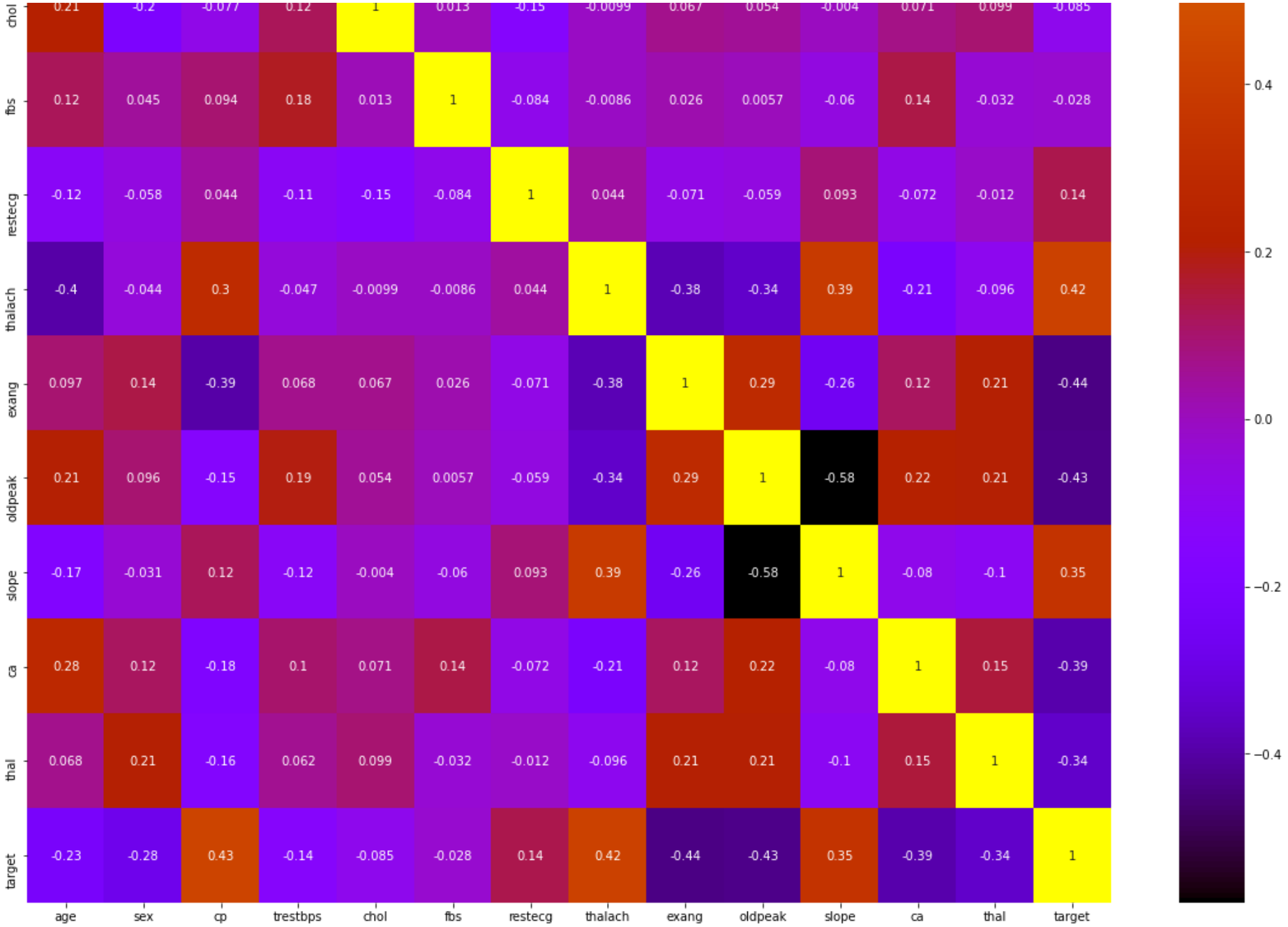
Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope

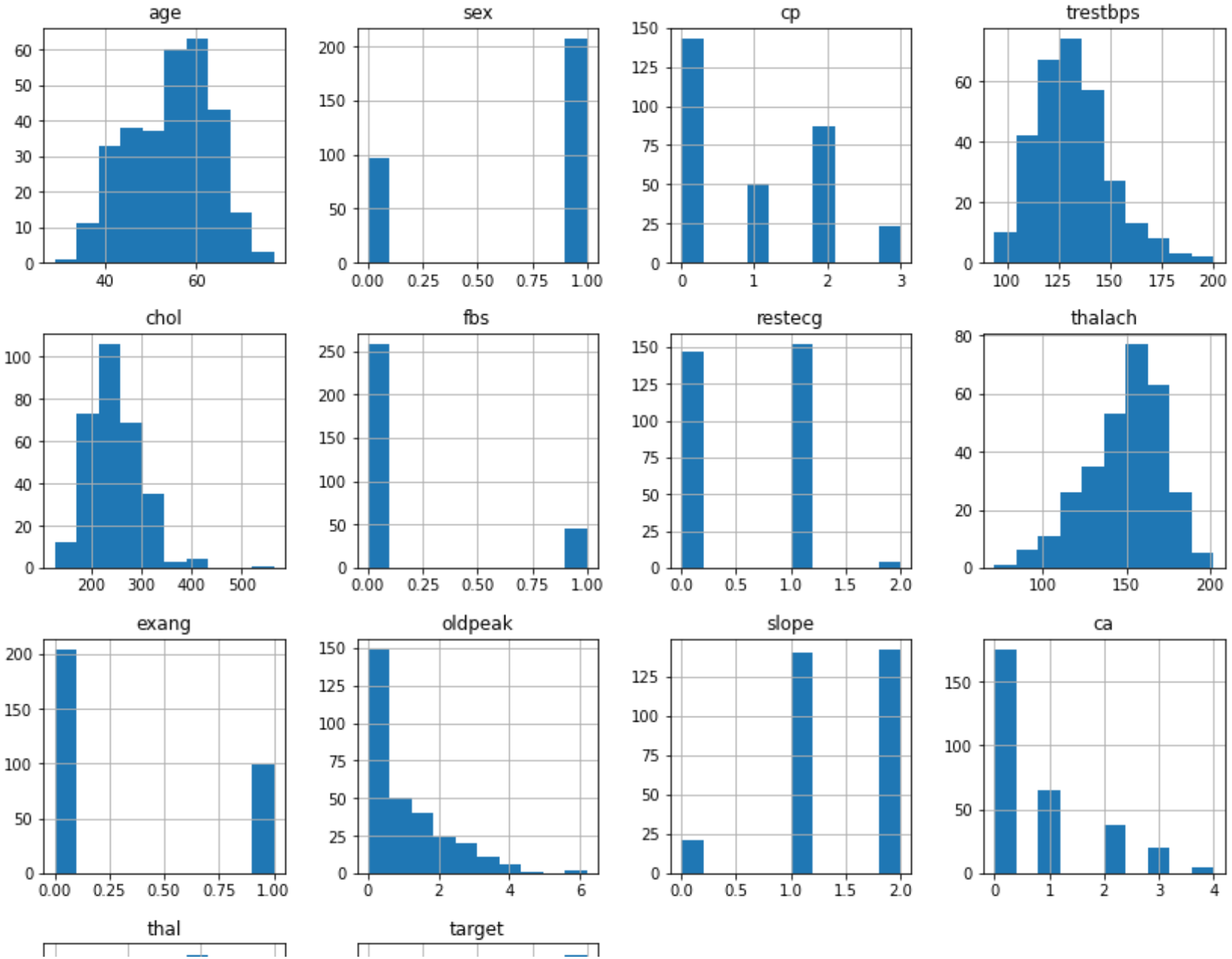
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.72
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.02
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.00
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.00
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.00
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.00
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.00

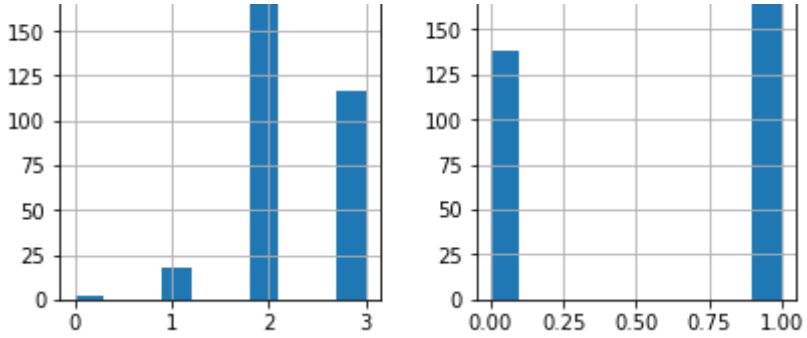
In [32]:
```
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="gnuplot")
```



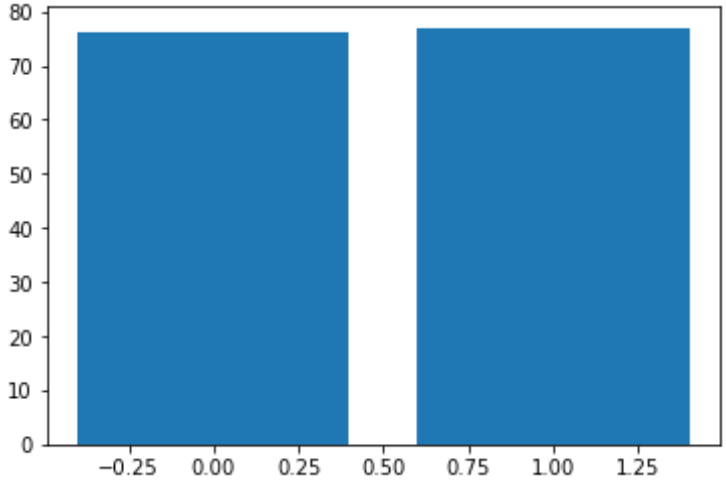


In [6]:
```
df.hist(figsize=(14,14))
plt.show()
```





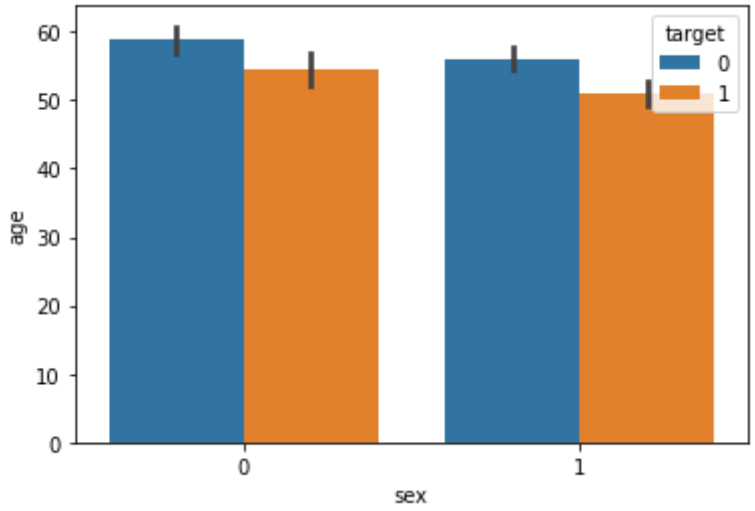
In [7]:
```
plt.bar(x=df['sex'],height=df['age'])
plt.show()
```



In [10]:
```
import seaborn as sns
```

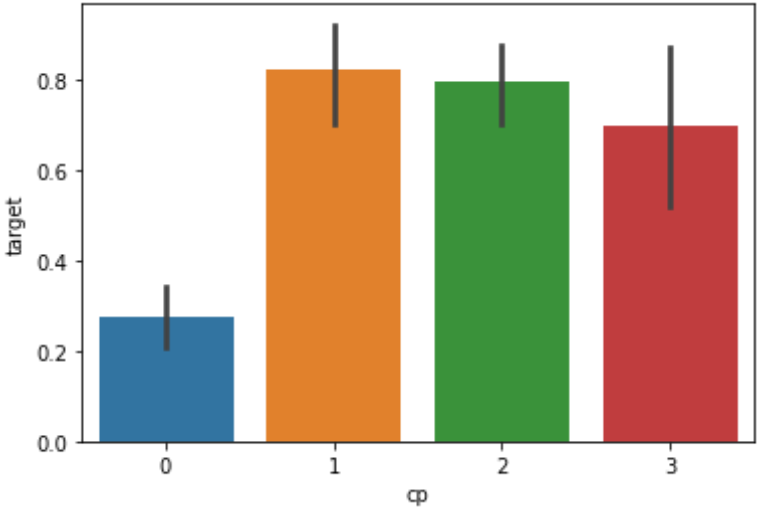
In [11]:
```
sns.barplot(x=df['sex'],y=df['age'],hue=df['target'])
```

Out[11]:
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



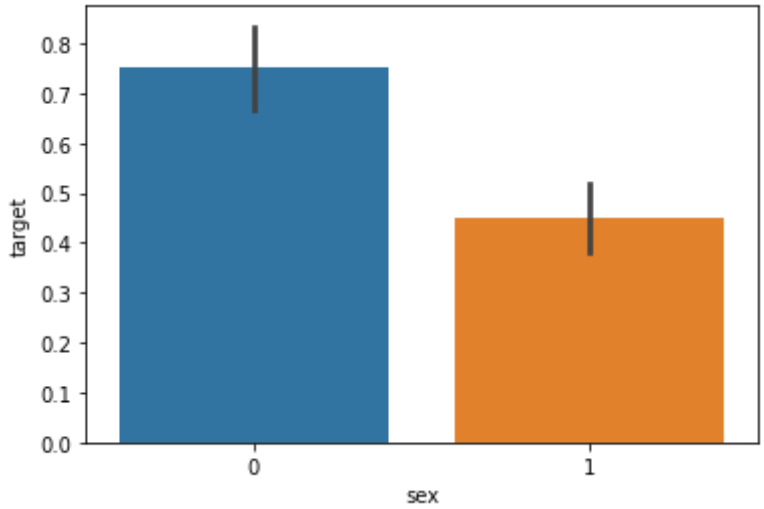
In [12]:
```
sns.barplot(df["cp"],df['target'])
```

Out[12]: <AxesSubplot:xlabel='cp', ylabel='target'>



In [13]:
```
sns.barplot(df["sex"],df['target'])
```

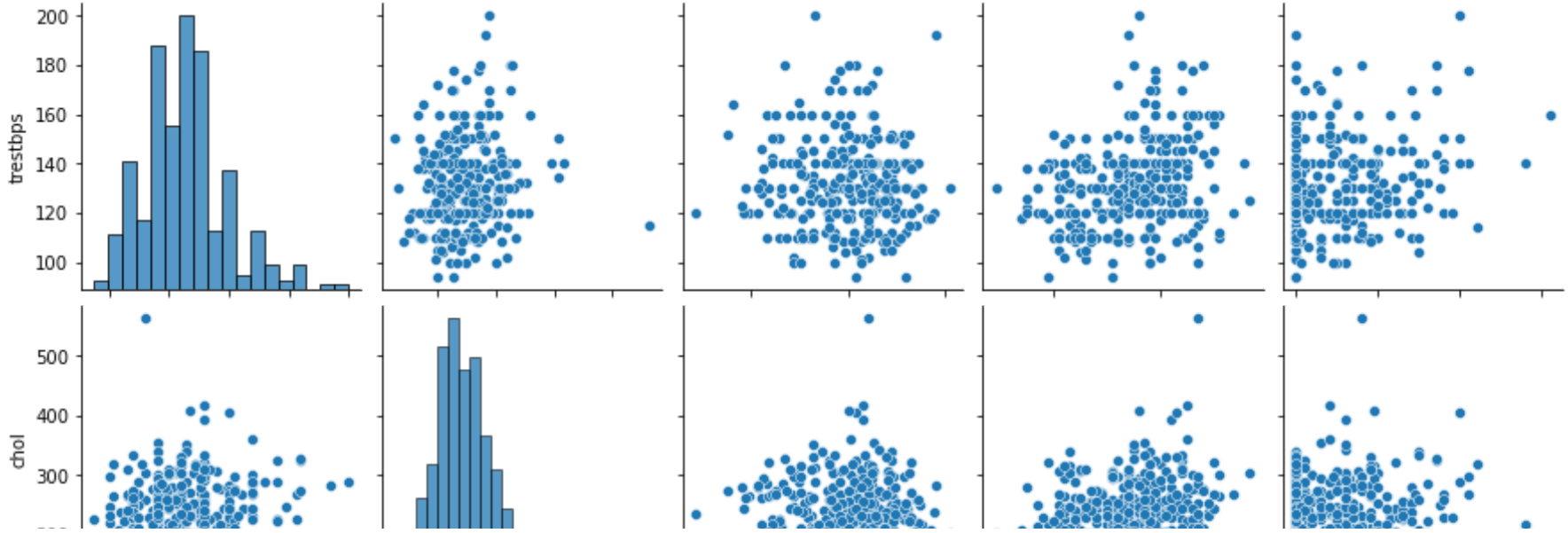
Out[13]: <AxesSubplot:xlabel='sex', ylabel='target'>

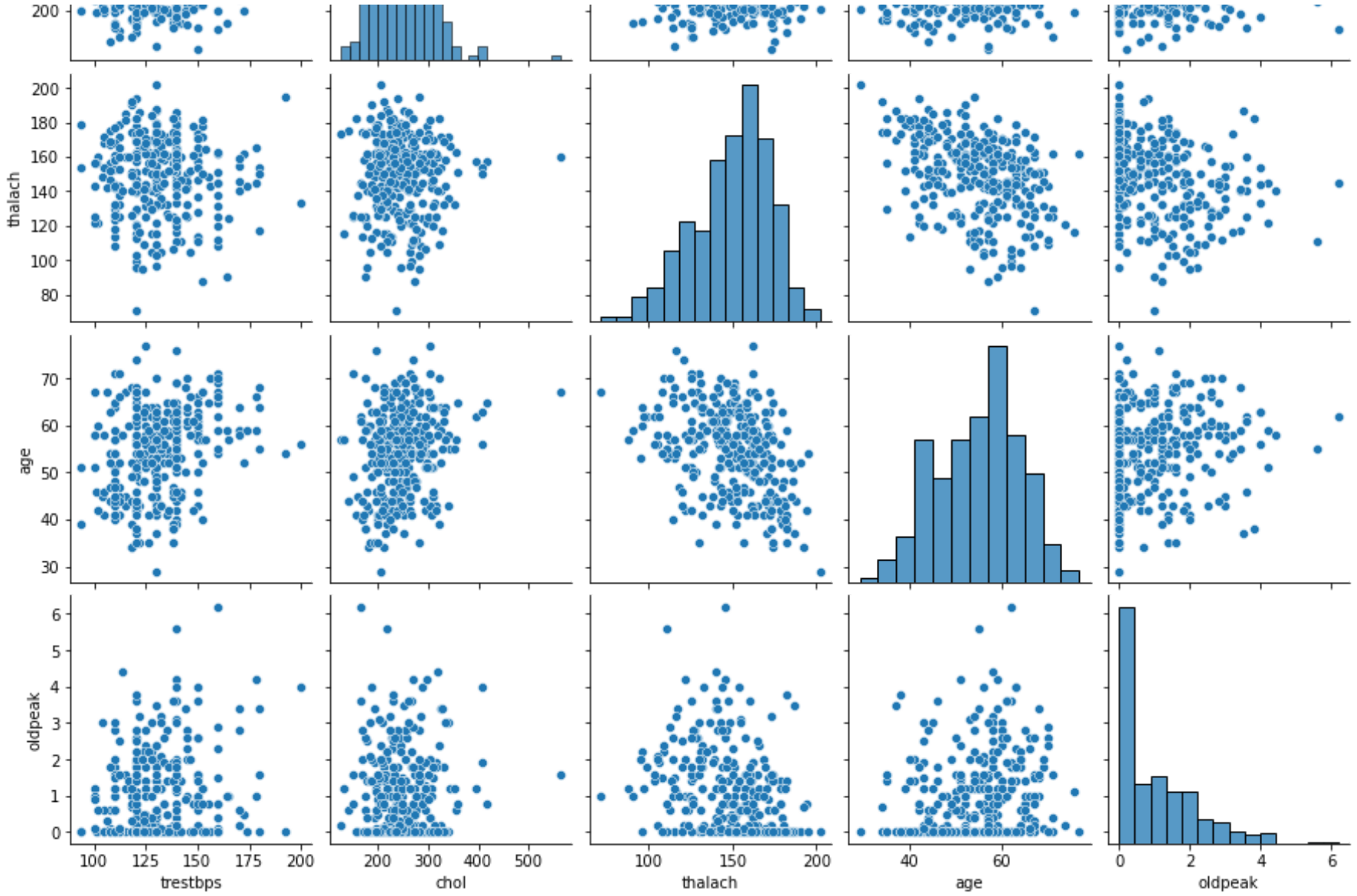


In [17]:
```
num_col=['trestbps','chol','thalach','age','oldpeak']
```

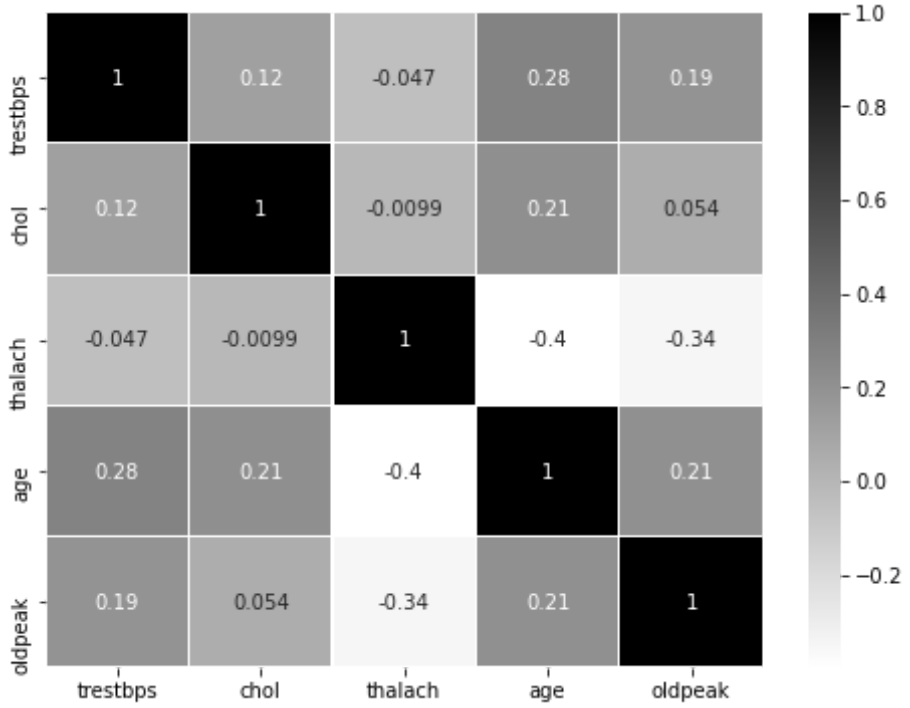
In [18]:
```
sns.pairplot(df[num_col])
```

Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x231a2663640>
```





In [33]:
```
sns.heatmap(df[num_col].corr(),annot=True, cmap='binary', linewidths=0.1)
fig=plt.gcf()
fig.set_size_inches(8,6)
plt.show()
```



In [34]:
```
df['target'].value_counts()
```

Out[34]:
```
1    165
0    138
Name: target, dtype: int64
```

In [35]:
```
df['target'].isnull()
```

Out[35]:
```
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 [37]:
```
X,y=df,df.target
```

In [38]:
```
X.drop('target',axis=1,inplace=True)
```

In [40]:
```
y.shape
```

Out[40]:
```
(303,)
```

In [41]:
```
X.shape
```

Out[41]:
```
(303, 13)
```

In [42]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

In [43]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=10,test_size=0.3,shuffle=True)
```

In [46]:
```
scores_dict = {}
```

In [49]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,mean_squared_error
dt=DecisionTreeClassifier()
dt.fit(X_train,y_train)
```

Out[49]:
```
DecisionTreeClassifier()
```

In [50]:
```
prediction=dt.predict(X_test)
accuracy_dt=accuracy_score(y_test,prediction)*100
```

In [51]:
```
scores_dict['DecisionTreeClassifier'] = accuracy_dt
print(accuracy_dt)
```

```
74.72527472527473
```

In [53]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
```

Out[53]:
```
KNeighborsClassifier()
```

In [54]:
```
prediction_knn=knn.predict(X_test)
accuracy_knn=accuracy_score(y_test,prediction_knn)*100
print('accuracy_score score     : ',accuracy_score(y_test,prediction_knn)*100,'%')
print('mean_squared_error score : ',mean_squared_error(y_test,prediction_knn)*100,'%')
```

```
accuracy_score score     :  81.31868131868131 %
mean_squared_error score :  18.681318681318682 %
```

In [55]:
```
scores_dict['KNeighborsClassifier'] = accuracy_knn
accuracy_knn
```

Out[55]:
```
81.31868131868131
```

In [56]:
```
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: 0.863
Accuracy on test set: 0.813
```

In [63]:
```
from sklearn import svm
clf=svm.SVC(kernel='rbf')
clf.fit(X_train,y_train)
Y_pred=clf.predict(X_test)
(accuracy_score(Y_pred,y_test)*100)
```

Out[63]:
```
80.21978021978022
```

In [64]:
```
from sklearn.linear_model import LogisticRegression
```

In [65]:
```
logmodel=LogisticRegression()
```

In [66]:
```
logmodel.fit(X_train, y_train)
```

Out[66]:
```
LogisticRegression()
```

In [67]:
```
predict = logmodel.predict(X_test)
```

In [68]:
```
from sklearn.metrics import classification_report
classification_report(y_test,predict)
```

Out[68]:
```
'              precision    recall  f1-score   support\n\n           0       0.89      0.66      0.76        50\n
1       0.69      0.90      0.78        41\n\n    accuracy                           0.77        91\n   macro avg
0.79      0.78      0.77        91\nweighted avg       0.80      0.77      0.77        91\n'
```

In [69]:
```
accuracy_score(y_test,predict)
```

Out[69]:
```
0.7692307692307693
```

In [ ]:
```

```