# program2-heart

March 29, 2024

[241]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set_theme(color_codes = True)
```

[242]:
```
df = pd.read_csv('HeartDisease.csv')
df
```

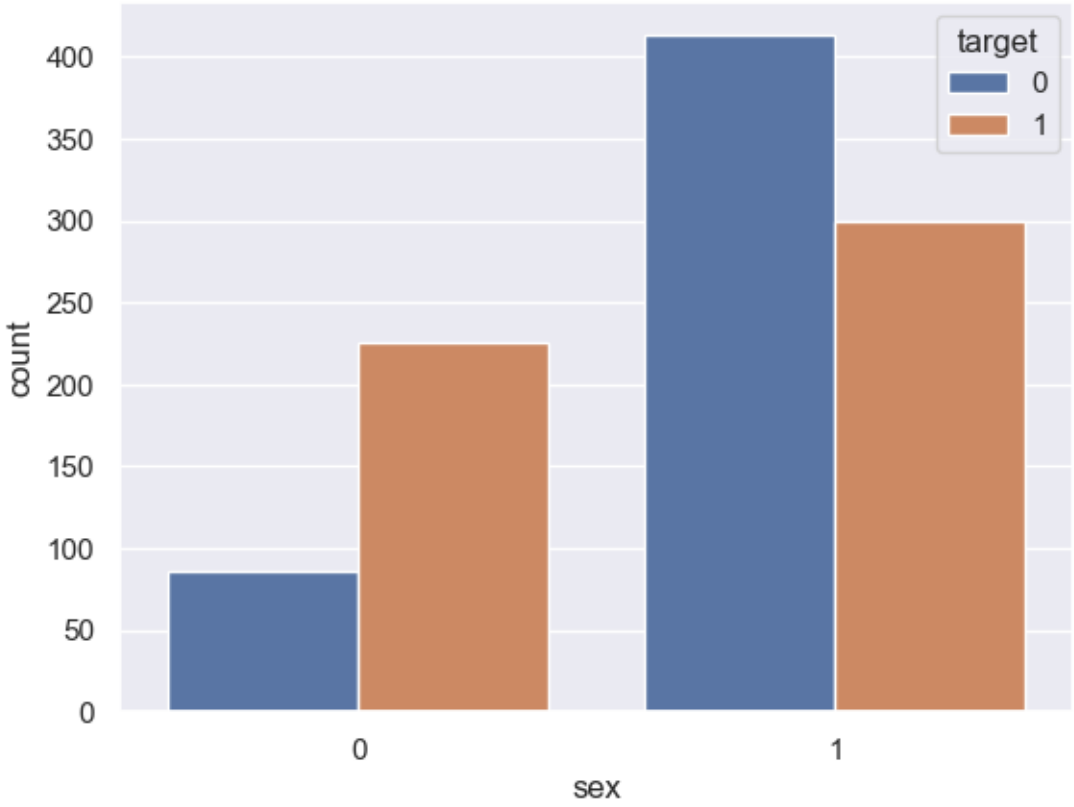
[242]:

| | 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 |
| … | … | … | .. | … | … | … | … | … | … | … | … | .. | … | … |
| 1020 | 59 | 1 | 1 | 140 | 221 | 0 | 1 | 164 | 1 | 0.0 | 2 | 0 | 2 | 1 |
| 1021 | 60 | 1 | 0 | 125 | 258 | 0 | 0 | 141 | 1 | 2.8 | 1 | 1 | 3 | 0 |
| 1022 | 47 | 1 | 0 | 110 | 275 | 0 | 0 | 118 | 1 | 1.0 | 1 | 1 | 2 | 0 |
| 1023 | 50 | 0 | 0 | 110 | 254 | 0 | 0 | 159 | 0 | 0.0 | 2 | 0 | 2 | 1 |
| 1024 | 54 | 1 | 0 | 120 | 188 | 0 | 1 | 113 | 0 | 1.4 | 1 | 1 | 3 | 0 |

[1025 rows x 14 columns]

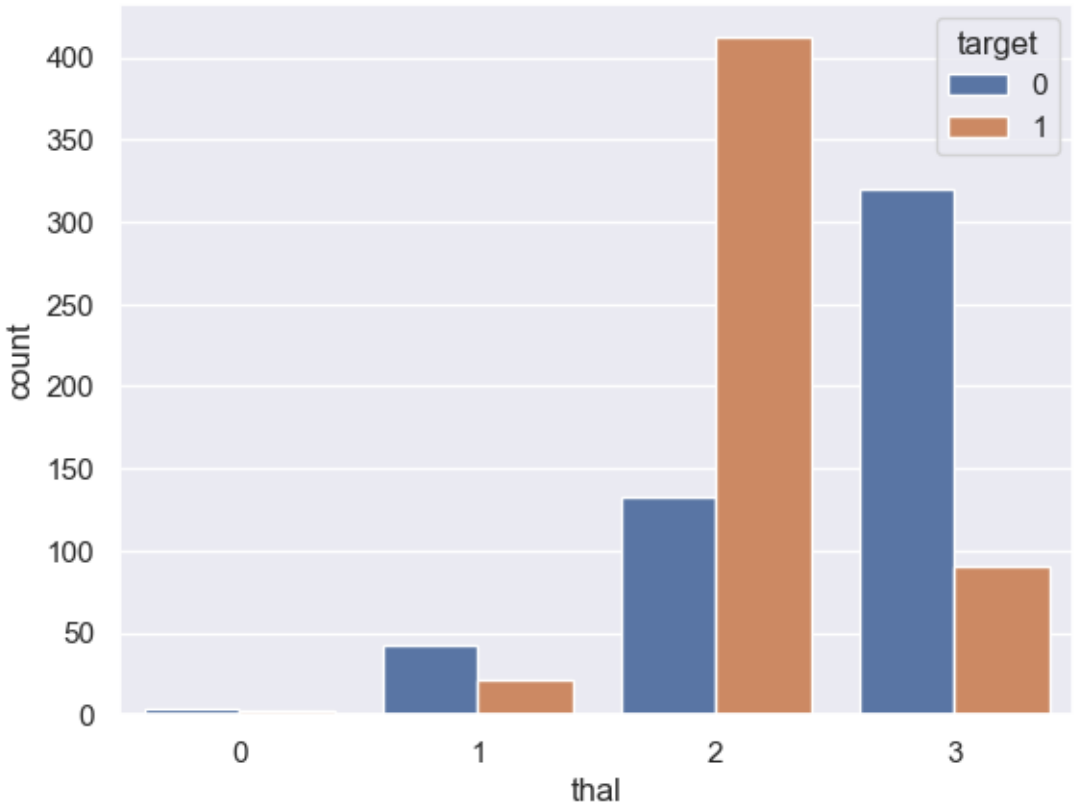
[243]:
```
sns.countplot(data=df, x="sex", hue="target")
```

[243]: <Axes: xlabel='sex', ylabel='count'>



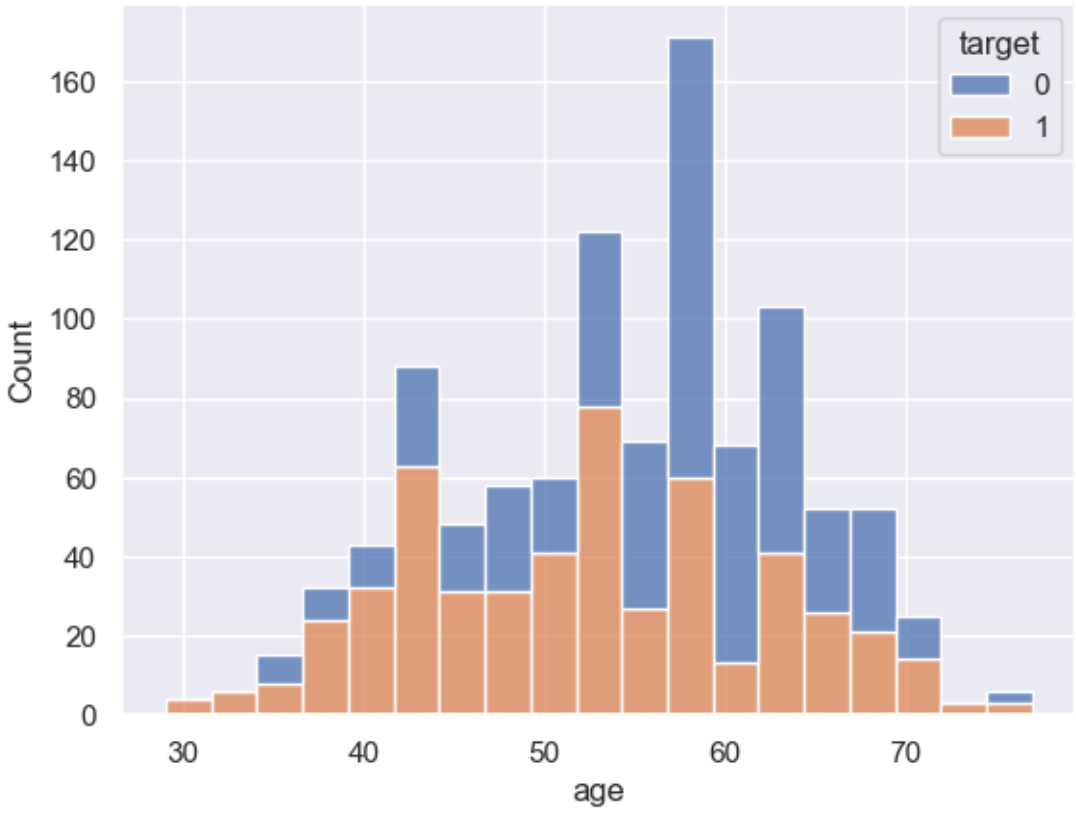

[244]:
```
sns.countplot(data=df, x="thal", hue="target")
```

[244]: <Axes: xlabel='thal', ylabel='count'>



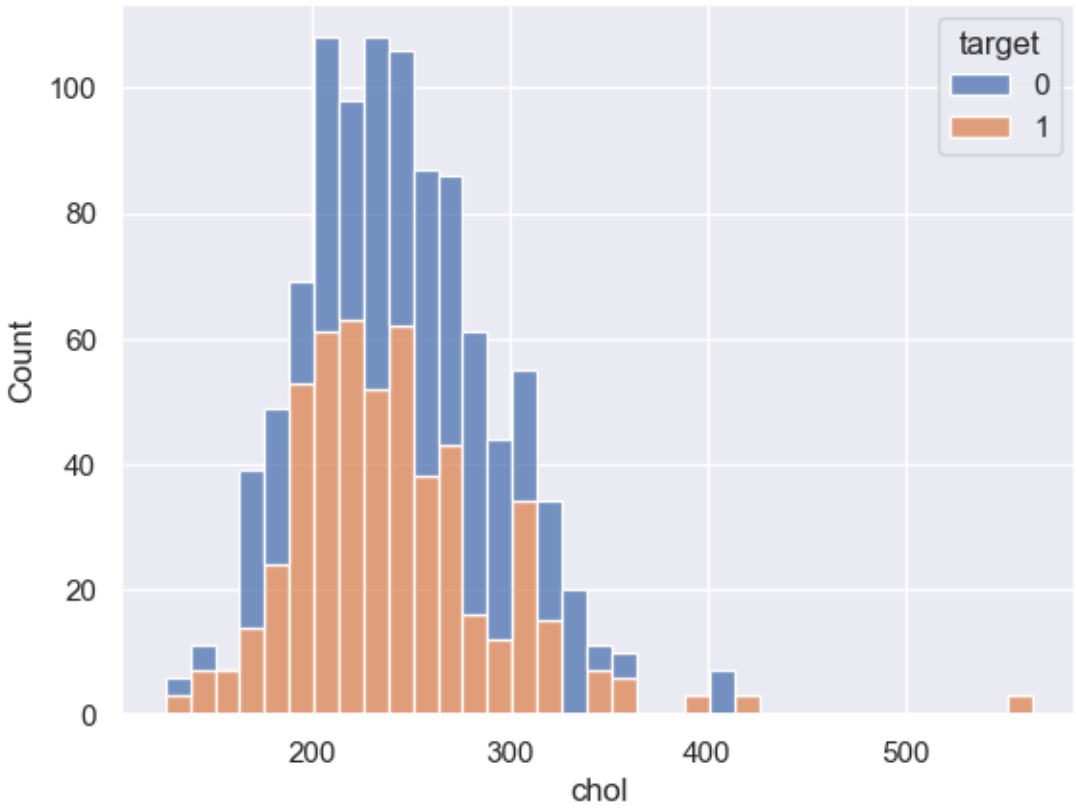

[245]:
```
sns.histplot(data=df, x="age", hue="target", multiple="stack")
```

[245]: <Axes: xlabel='age', ylabel='Count'>



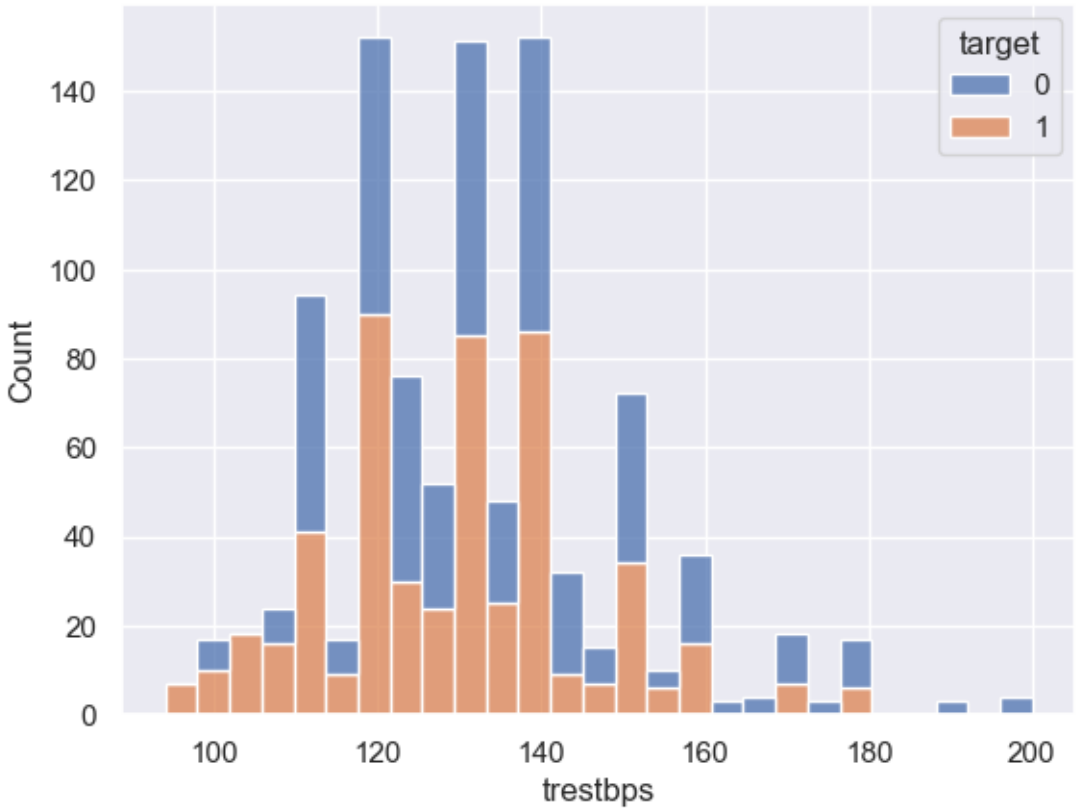

[246]:
```
sns.histplot(data=df, x="chol", hue="target", multiple="stack")
```

[246]:
```
<Axes: xlabel='chol', ylabel='Count'>
```



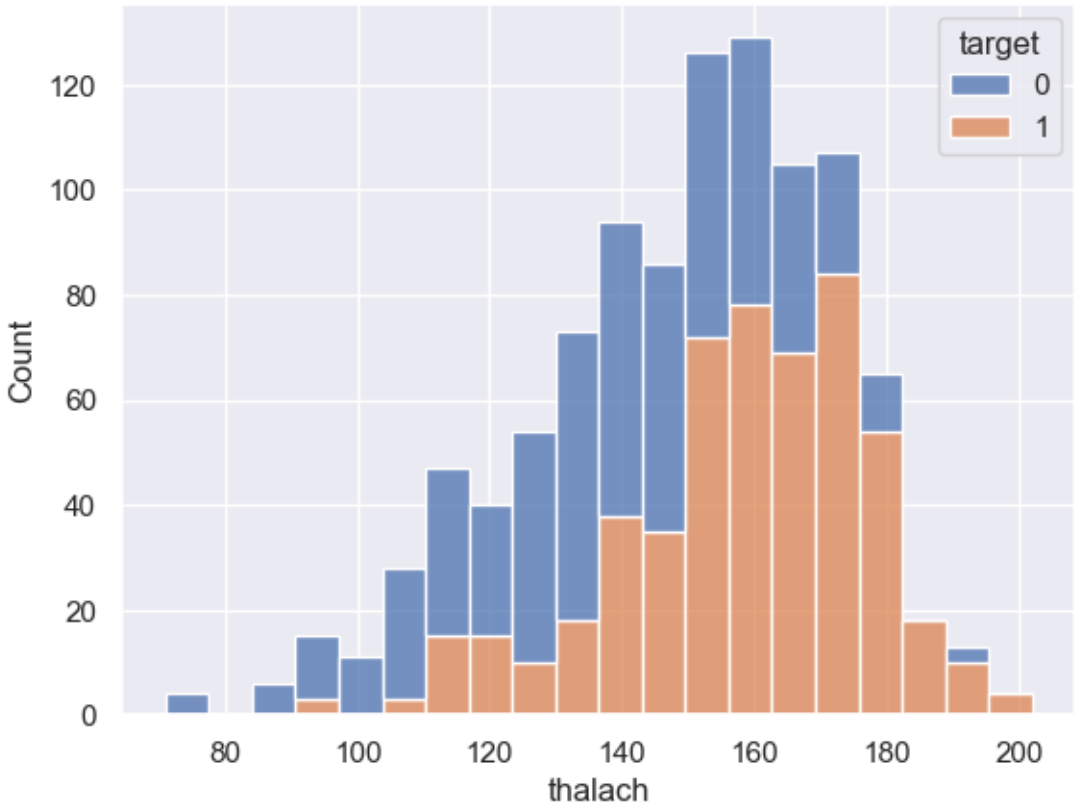

[247]:
```
sns.histplot(data=df, x="trestbps", hue="target", multiple="stack")
```

[247]:
```
<Axes: xlabel='trestbps', ylabel='Count'>
```




[248]:
```
sns.histplot(data=df, x="thalach", hue="target", multiple="stack")
```

[248]:
```
<Axes: xlabel='thalach', ylabel='Count'>
```

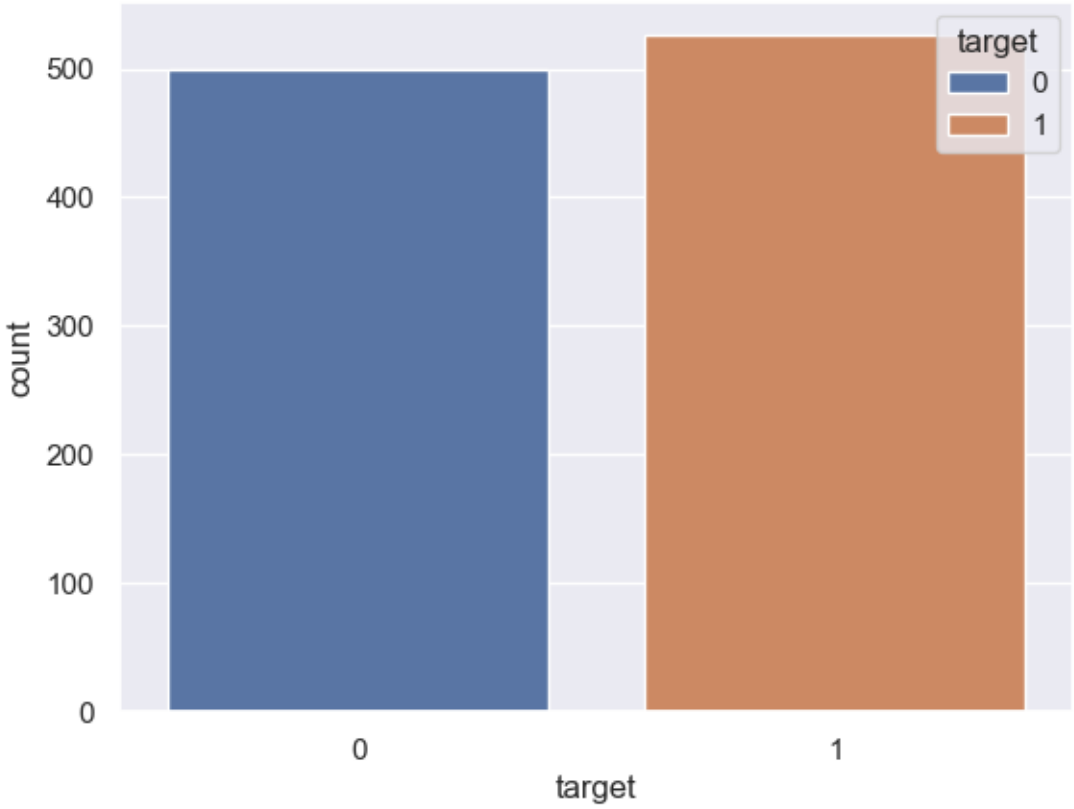



[249]:
```
df.isnull().sum()
```

[249]:
```
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
```

[250]:
```
sns.countplot(data = df,x='target',hue="target")
print(df.target.value_counts())
```

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

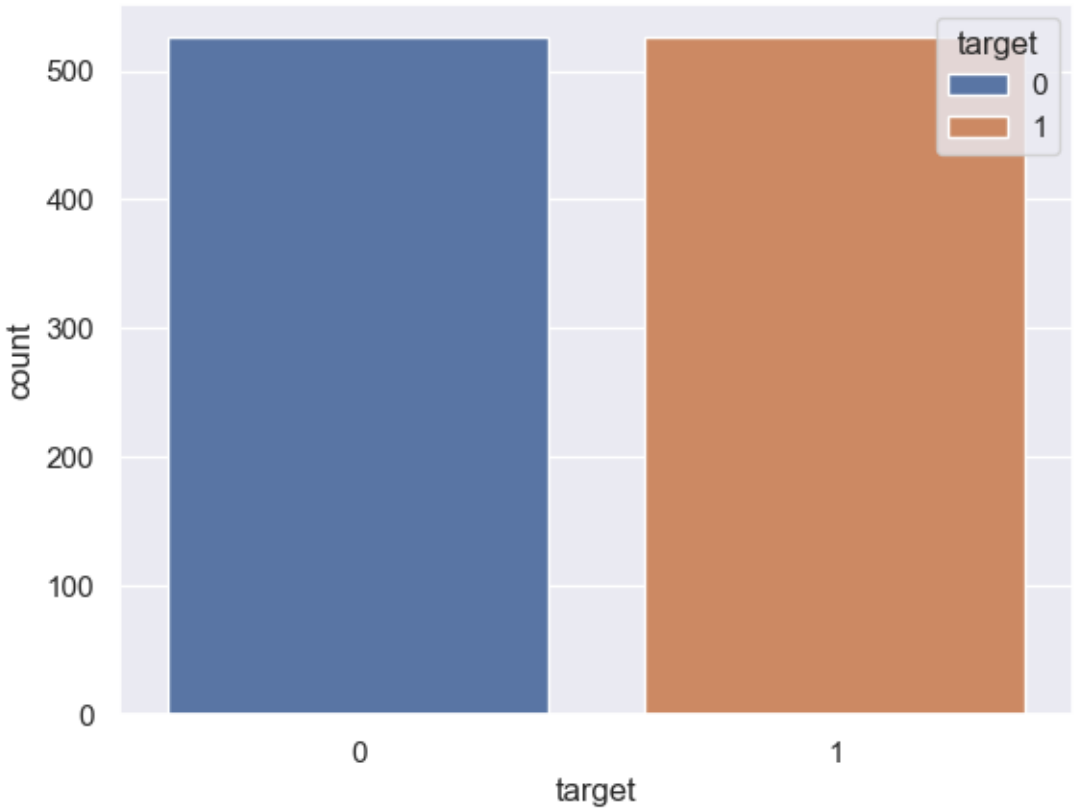



[251]:
```
from sklearn.utils import resample
df_maj=df[(df['target']==1)]
df_min=df[(df['target']==0)]
df_min_up=resample(df_min,
                   n_samples=526,
                   random_state=0)
```

[252]:
```
df2=pd.concat([df_min_up,df_maj])
```

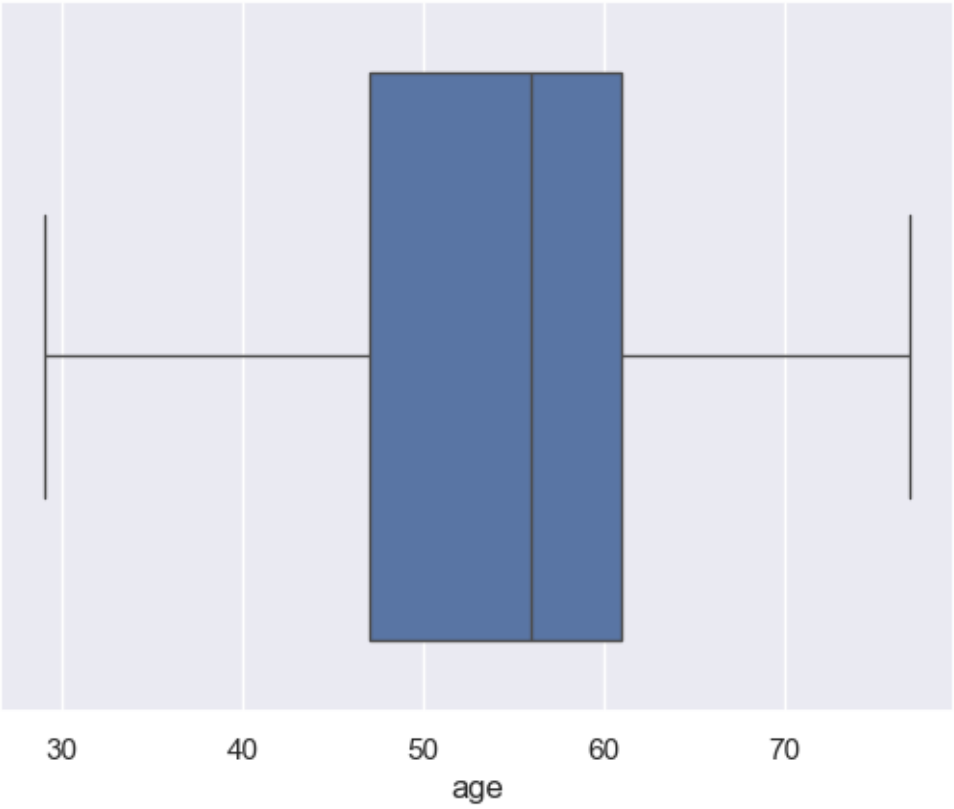
[253]:
```
sns.countplot(data = df2,x='target',hue="target")
print(df2.target.value_counts())
```

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



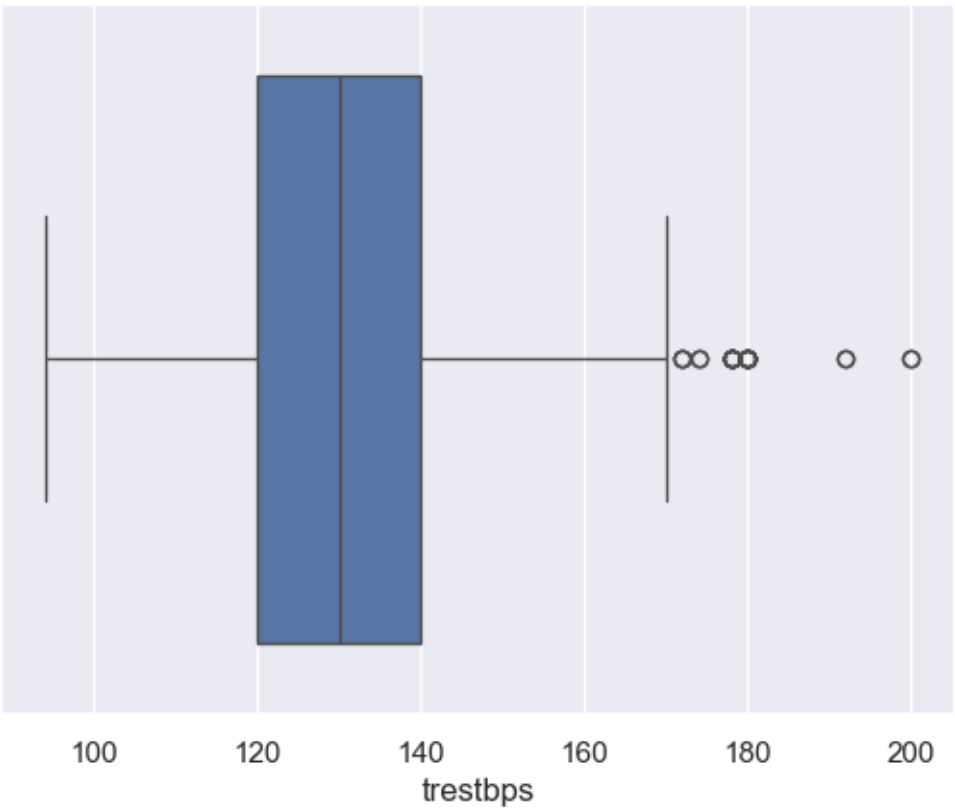

[254]:
```
sns.boxplot(x=df2["age"])
```

[254]:
```
<Axes: xlabel='age'>
```



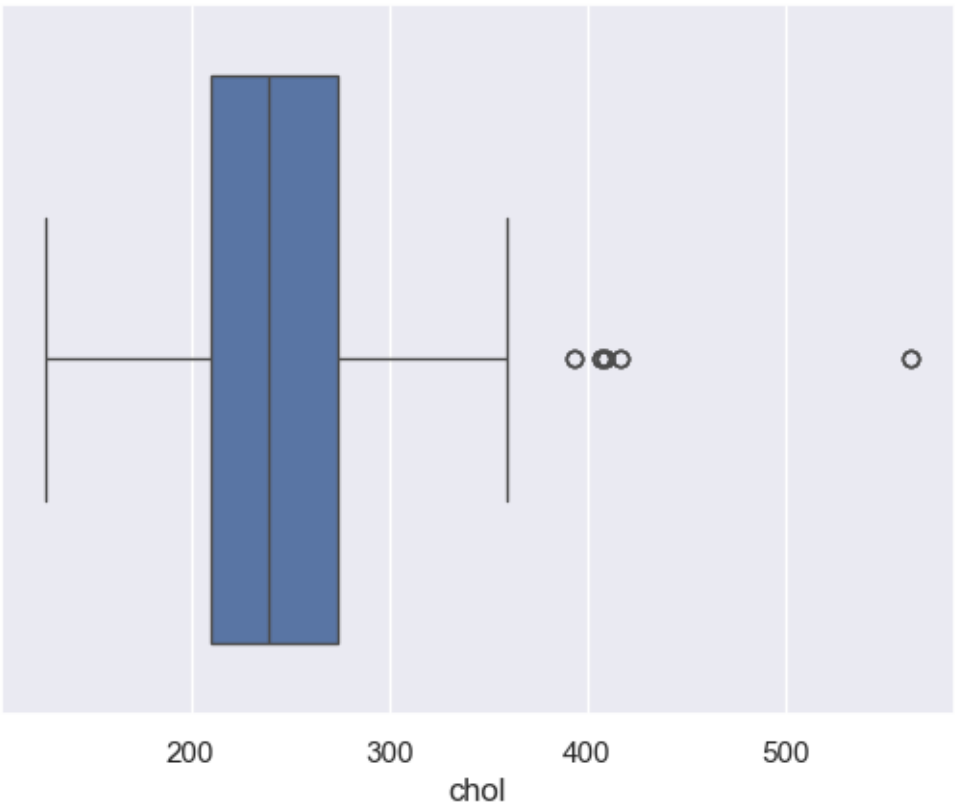

[255]:
```
sns.boxplot(x=df2["trestbps"])
```

[255]:
```
<Axes: xlabel='trestbps'>
```



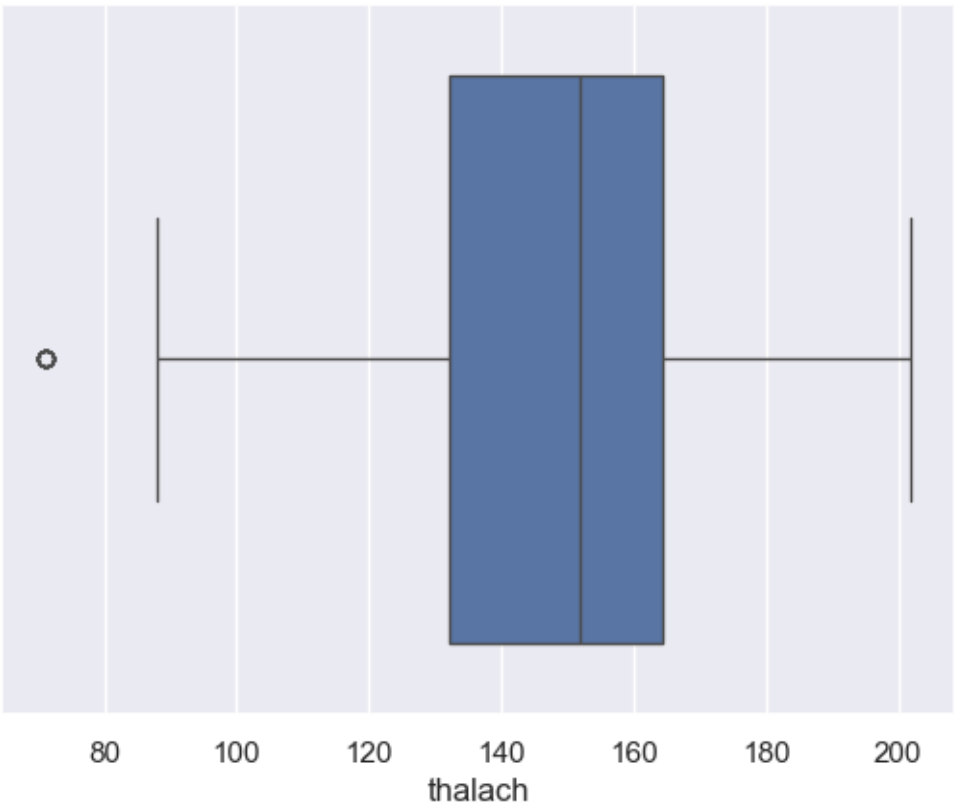

[256]:
```
sns.boxplot(x=df2["chol"])
```

[256]:
```
<Axes: xlabel='chol'>
```



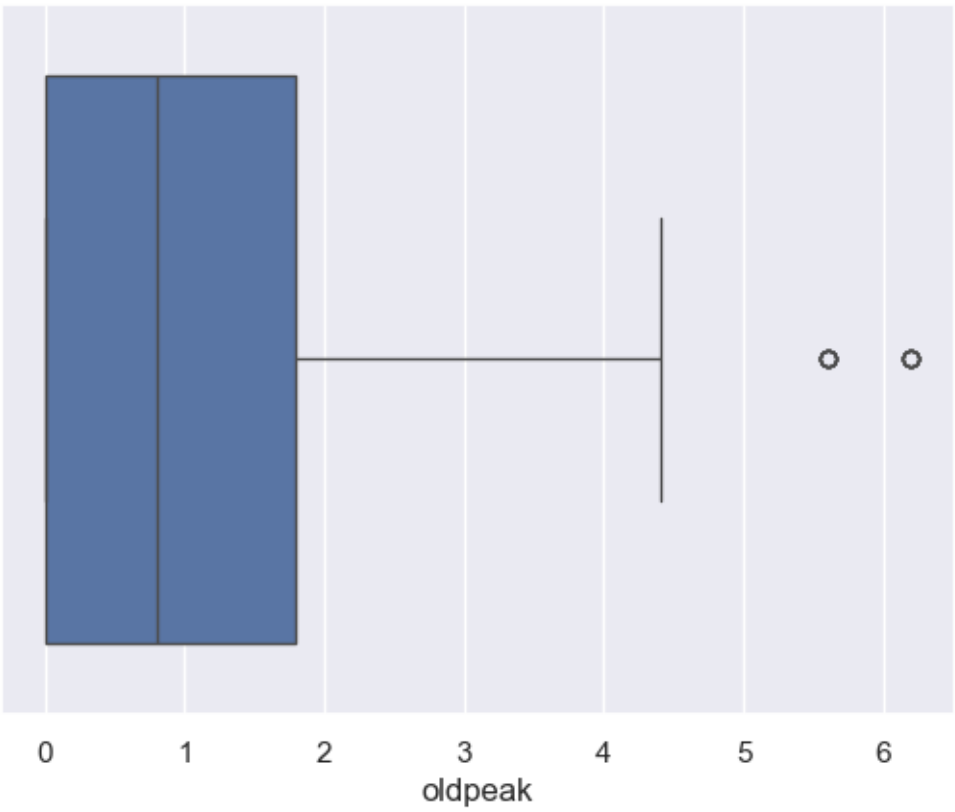

[257]:
```
sns.boxplot(x=df2["thalach"])
```

[257]:
```
<Axes: xlabel='thalach'>
```




[258]:
```
sns.boxplot(x=df2["oldpeak"])
```

[258]:
```
<Axes: xlabel='oldpeak'>
```

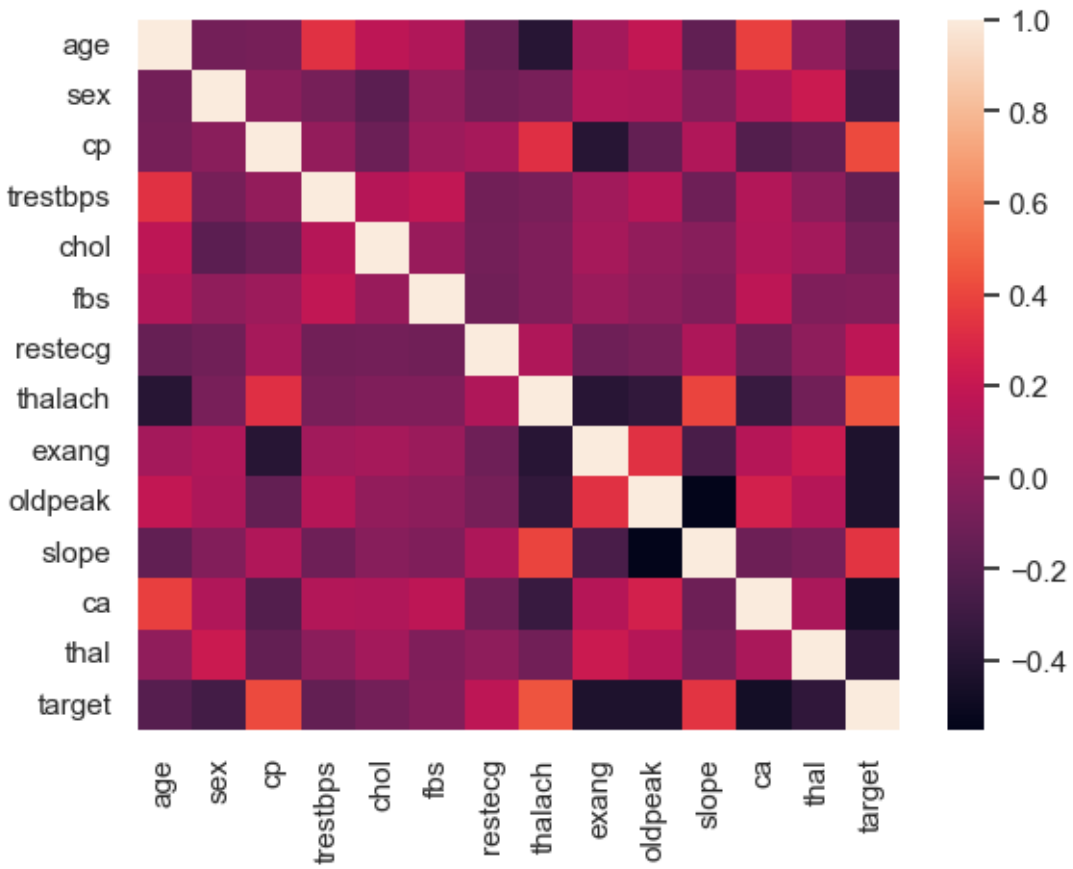



[259]:
```
import scipy.stats as stats
z = np.abs(stats.zscore(df2))
data_clean = df2[(z<3).all(axis = 1)]
data_clean.shape
```

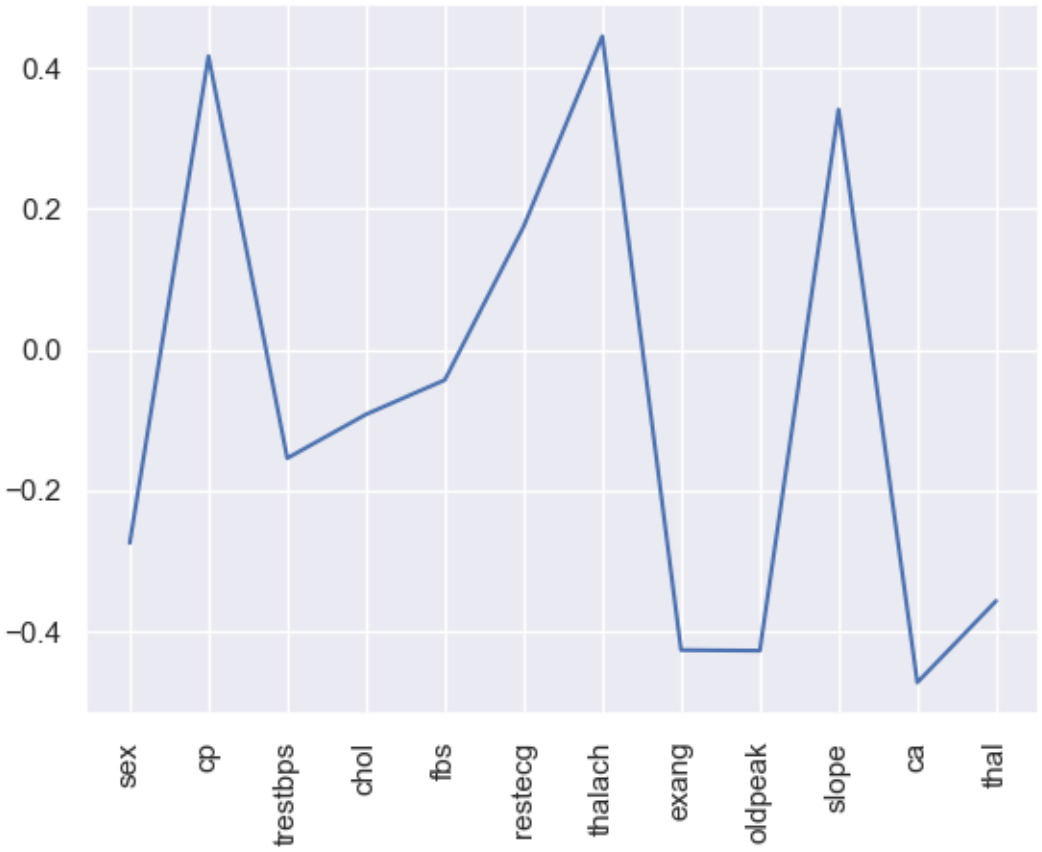
[259]: (989, 14)

[260]:
```
sns.heatmap(data_clean.corr(), fmt='.2g')
```

[260]: <Axes: >




[261]:
```
corr = data_clean[data_clean.columns[1:]].corr()['target'][:-1]
plt.plot(corr)
plt.xticks(rotation=90)
plt.show()
```




[262]:
```
X = data_clean.drop('target', axis=1)
y = data_clean['target']
```

[263]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.
 ↪2,random_state=0)
```

[264]:
```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(random_state = 0)
dtree.fit(X_train, y_train)
```

[264]:
```
DecisionTreeClassifier(random_state=0)
```

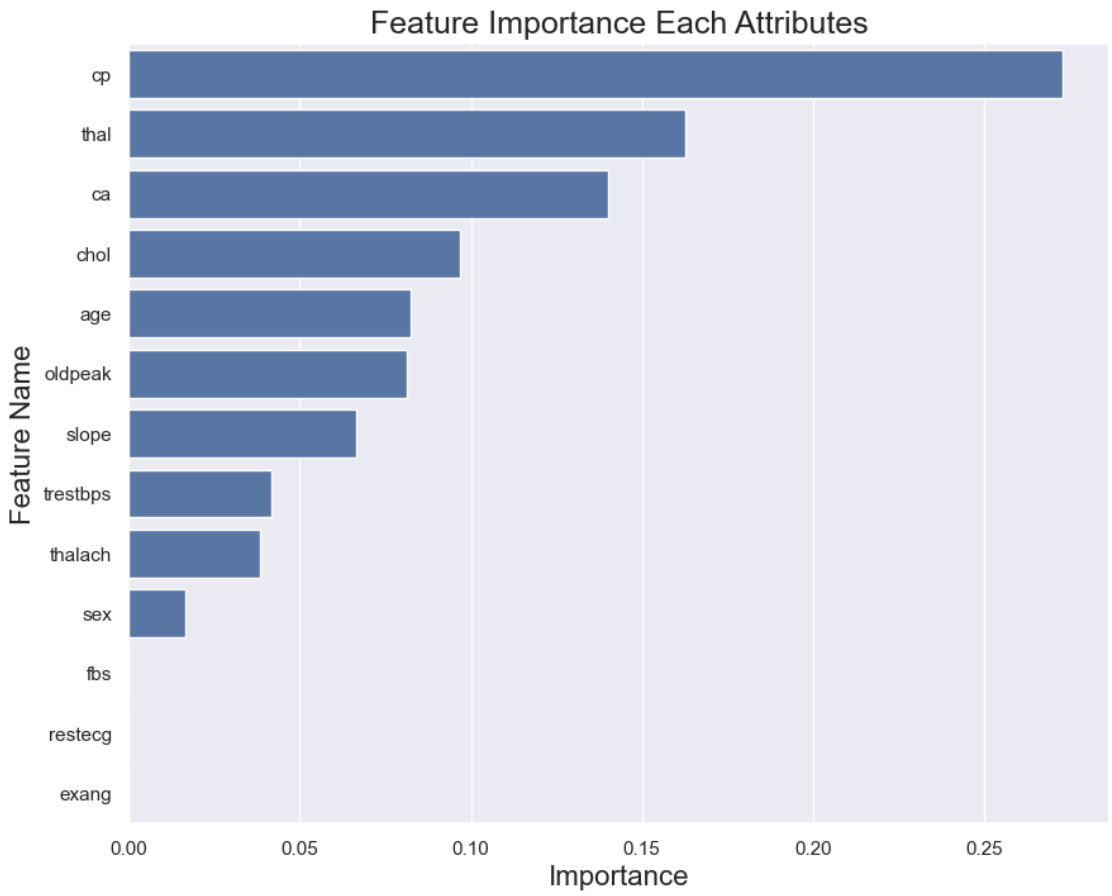
[265]:
```
y_pred = dtree.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 97.47 %
```

[266]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
 ↪recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.9765258215962441
Precision Score :  0.9811320754716981
Recall Score :  0.9719626168224299
```

[267]:
```
#Feature Importance
imp_df = pd.DataFrame({
    "Feature Name": X_train.columns,
    "Importance": dtree.feature_importances_
})
fi = imp_df.sort_values(by="Importance", ascending=False)
plt.figure(figsize=(10,8))
sns.barplot(data=fi, x='Importance', y='Feature Name')
plt.title('Feature Importance Each Attributes', fontsize=18)
plt.xlabel ('Importance', fontsize=16)
plt.ylabel ('Feature Name', fontsize=16)
plt.show()
```




[268]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state=0)
rfc.fit(X_train, y_train)
```

[268]:
```
RandomForestClassifier(random_state=0)
```

[269]:
```
y_pred = rfc.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

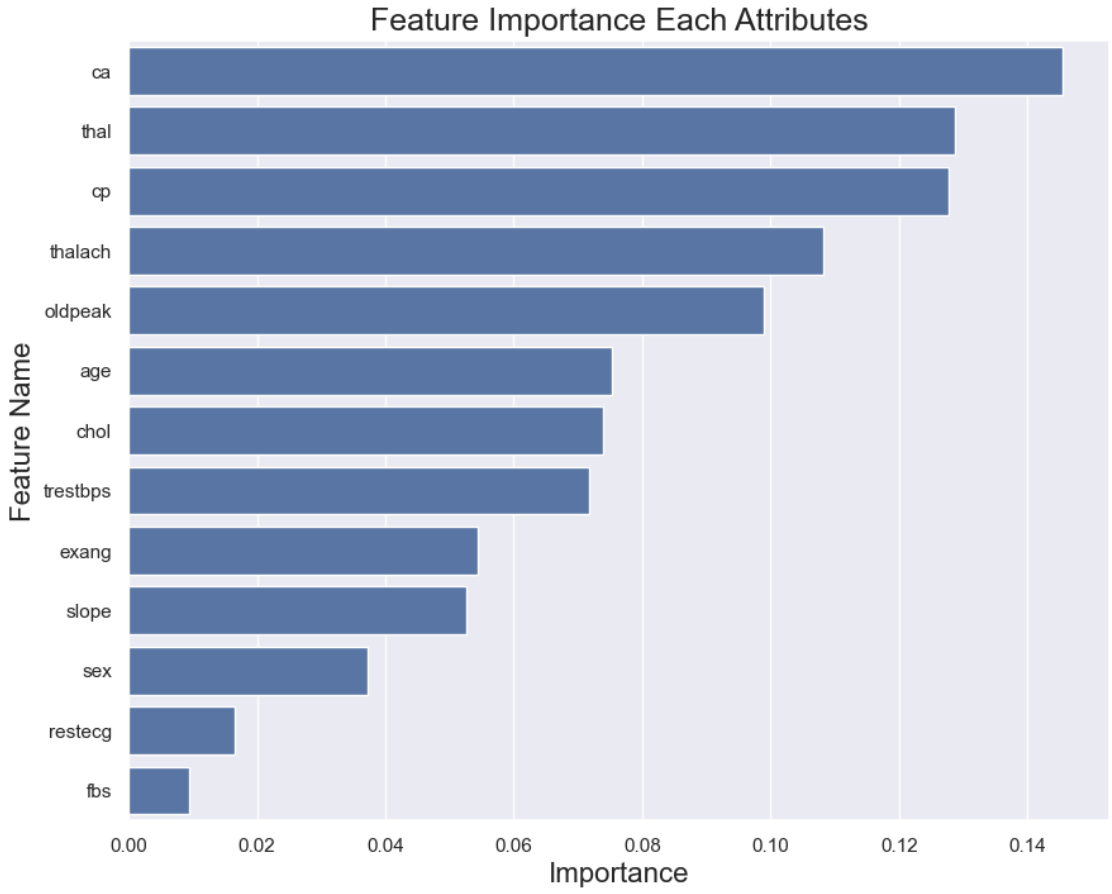
```
Accuracy Score : 97.47 %
```

[270]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
 ↪recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.9765258215962441
Precision Score :  0.9811320754716981
```

```
Recall Score :  0.9719626168224299
```

[271]:
```
#Feature Importance
imp_df = pd.DataFrame({
    "Feature Name": X_train.columns,
    "Importance": rfc.feature_importances_
})
fi = imp_df.sort_values(by="Importance", ascending=False)
plt.figure(figsize=(10,8))
sns.barplot(data=fi, x='Importance', y='Feature Name')
plt.title('Feature Importance Each Attributes', fontsize=18)
plt.xlabel ('Importance', fontsize=16)
plt.ylabel ('Feature Name', fontsize=16)
plt.show()
```




[272]:
```
from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(random_state=0)
ada.fit(X_train, y_train)
```

```
C:\Users\dell\AppData\Local\Programs\Python\Python312\Lib\site-
```

```
packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
```

[272]:
```
AdaBoostClassifier(random_state=0)
```

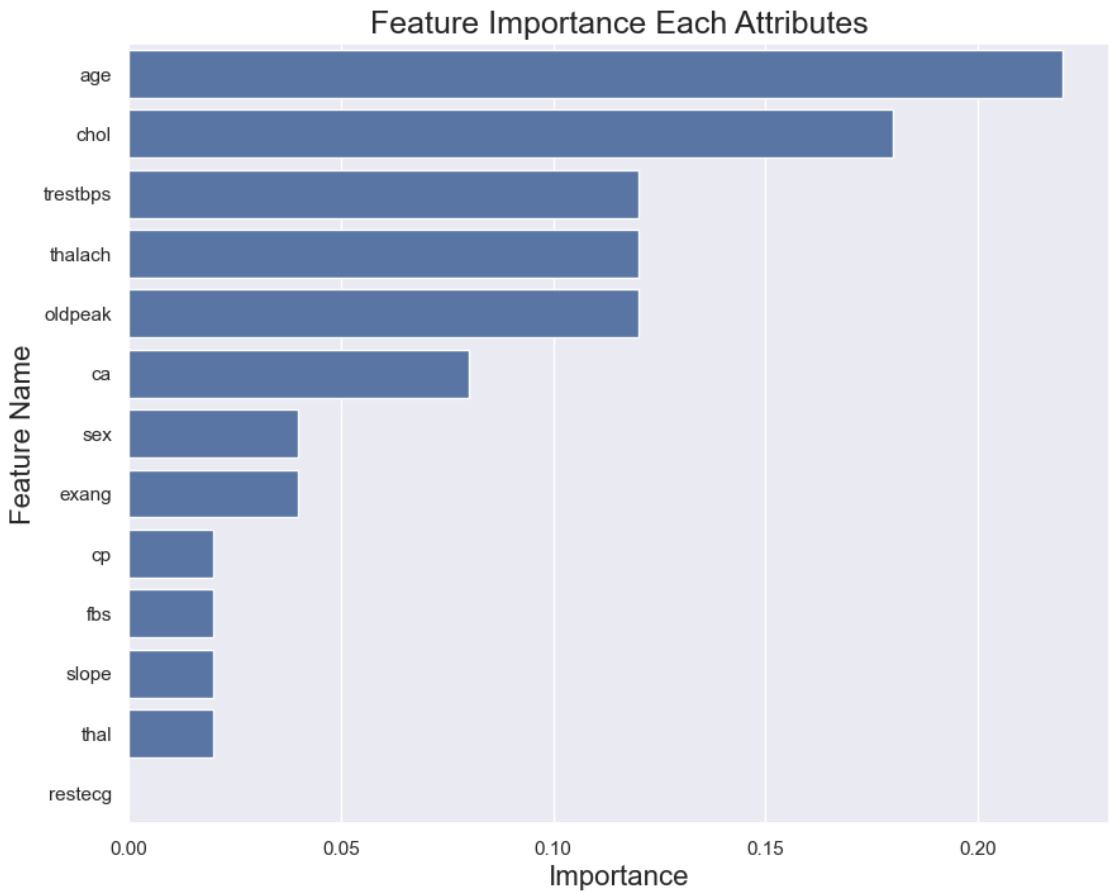
[273]:
```
y_pred = ada.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 87.37 %
```

[274]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
 ↪recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.8888888888888888
Precision Score :  0.847457627118644
Recall Score :  0.9345794392523364
```

[275]:
```
#Feature Importance
imp_df = pd.DataFrame({
    "Feature Name": X_train.columns,
    "Importance": ada.feature_importances_
})
fi = imp_df.sort_values(by="Importance", ascending=False)
plt.figure(figsize=(10,8))
sns.barplot(data=fi, x='Importance', y='Feature Name')
plt.title('Feature Importance Each Attributes', fontsize=18)
plt.xlabel ('Importance', fontsize=16)
plt.ylabel ('Feature Name', fontsize=16)
plt.show()
```


Feature Importance Each Attributes
age
chol
trestbps
thalach
oldpeak
ca
sex
exang
cp
fbs
slope
thal
restecg
Feature Name
0.00
0.05
0.10
0.15
0.20
Importance