arrythemia-prediction

[2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```python
data = pd.read_csv('heart.csv')
```

[4]:
```python
data.head()
```

[4]:
```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   3       145   233    1        0      150      0      2.3      0
1   37    1   2       130   250    0        1      187      0      3.5      0
2   41    0   1       130   204    0        0      172      0      1.4      2
3   56    1   1       120   236    0        1      178      0      0.8      2
4   57    0   0       120   354    0        1      163      1      0.6      2

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
```

[5]:
```python
data.shape
```

[5]: (303, 14)

[6]:
```python
data.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
```

[7]:
```
data.isnull().sum()
```

[7]:
```
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
```

[8]:
```
data.describe()
```

[8]:
```
              age         sex          cp    trestbps        chol         fbs  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean    54.366337    0.683168    0.966997  131.623762  246.264026    0.148515
std      9.082101    0.466011    1.032052   17.538143   51.830751    0.356198
min     29.000000    0.000000    0.000000   94.000000  126.000000    0.000000
25%     47.500000    0.000000    0.000000  120.000000  211.000000    0.000000
50%     55.000000    1.000000    1.000000  130.000000  240.000000    0.000000
75%     61.000000    1.000000    2.000000  140.000000  274.500000    0.000000
max     77.000000    1.000000    3.000000  200.000000  564.000000    1.000000

          restecg     thalach       exang     oldpeak       slope          ca  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean     0.528053  149.646865    0.326733    1.039604    1.399340    0.729373
std      0.525860   22.905161    0.469794    1.161075    0.616226    1.022606
```

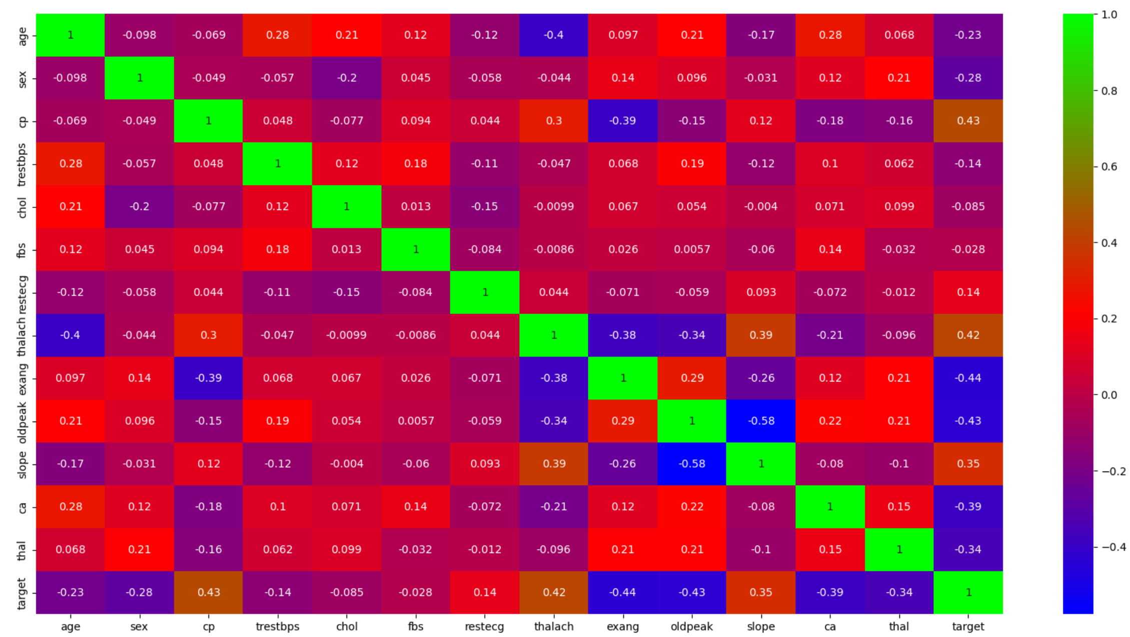
```
min       0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
25%       0.000000  133.500000    0.000000    0.000000    1.000000    0.000000
50%       1.000000  153.000000    0.000000    0.800000    1.000000    0.000000
75%       1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max       2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

             thal      target
count  303.000000  303.000000
mean     2.313531    0.544554
std      0.612277    0.498835
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

[9]:

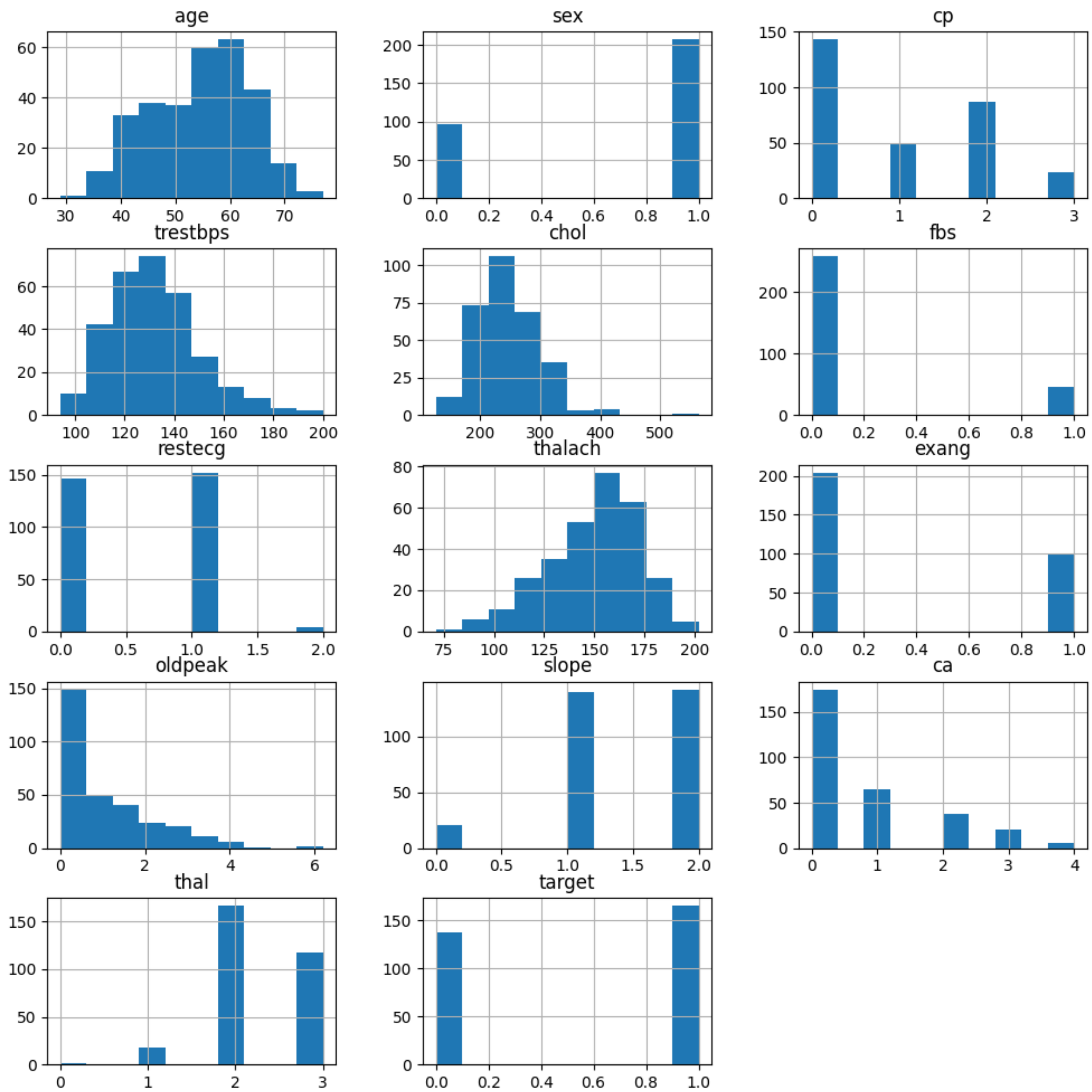
```python
plt.figure(figsize=(20,10))
sns.heatmap(data.corr(),annot=True,cmap="brg")
```

[9]: <Axes: >



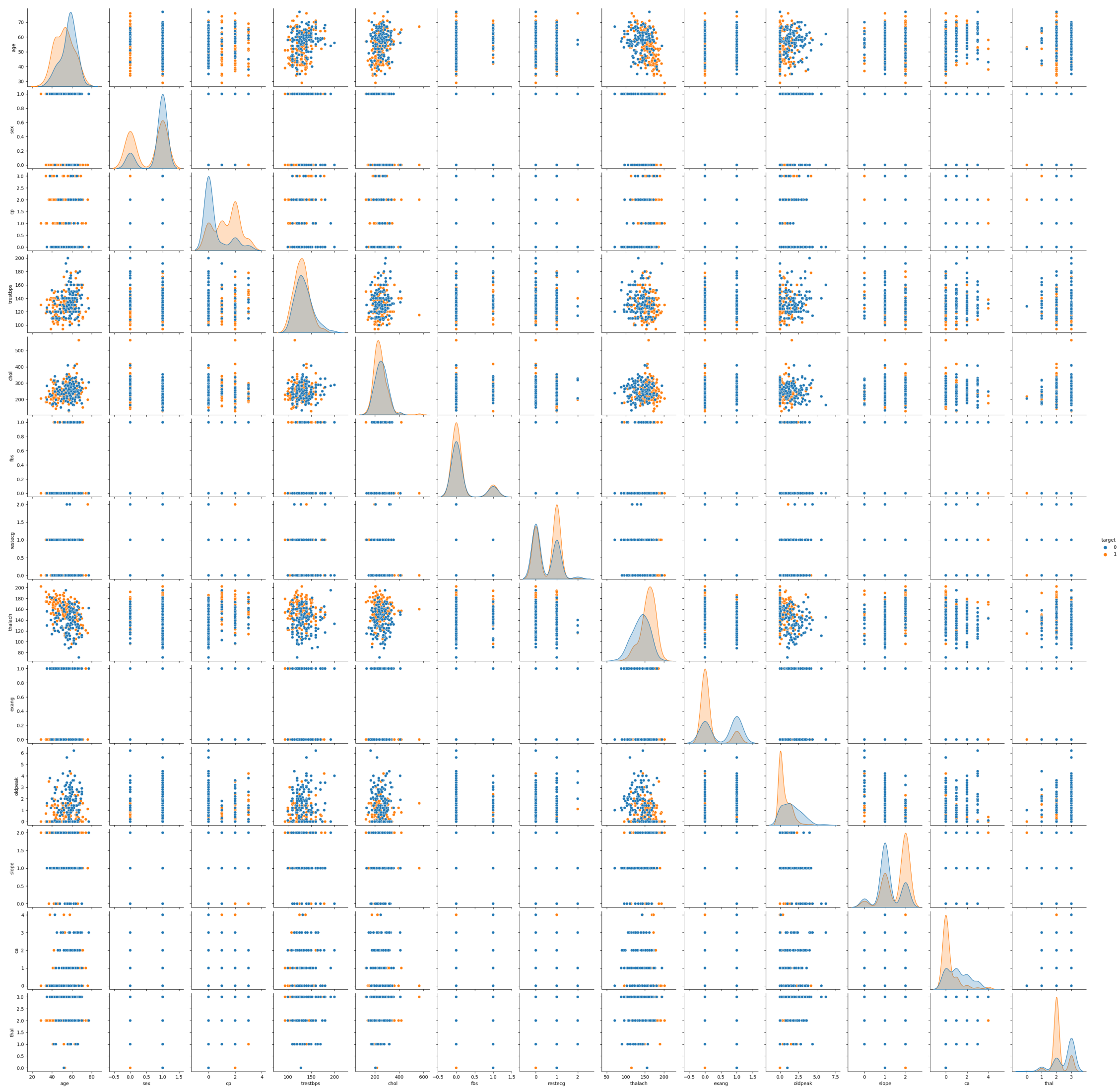
[10]:

```python
data.hist(figsize=(12,12),layout=(5,3))
plt.show()
```

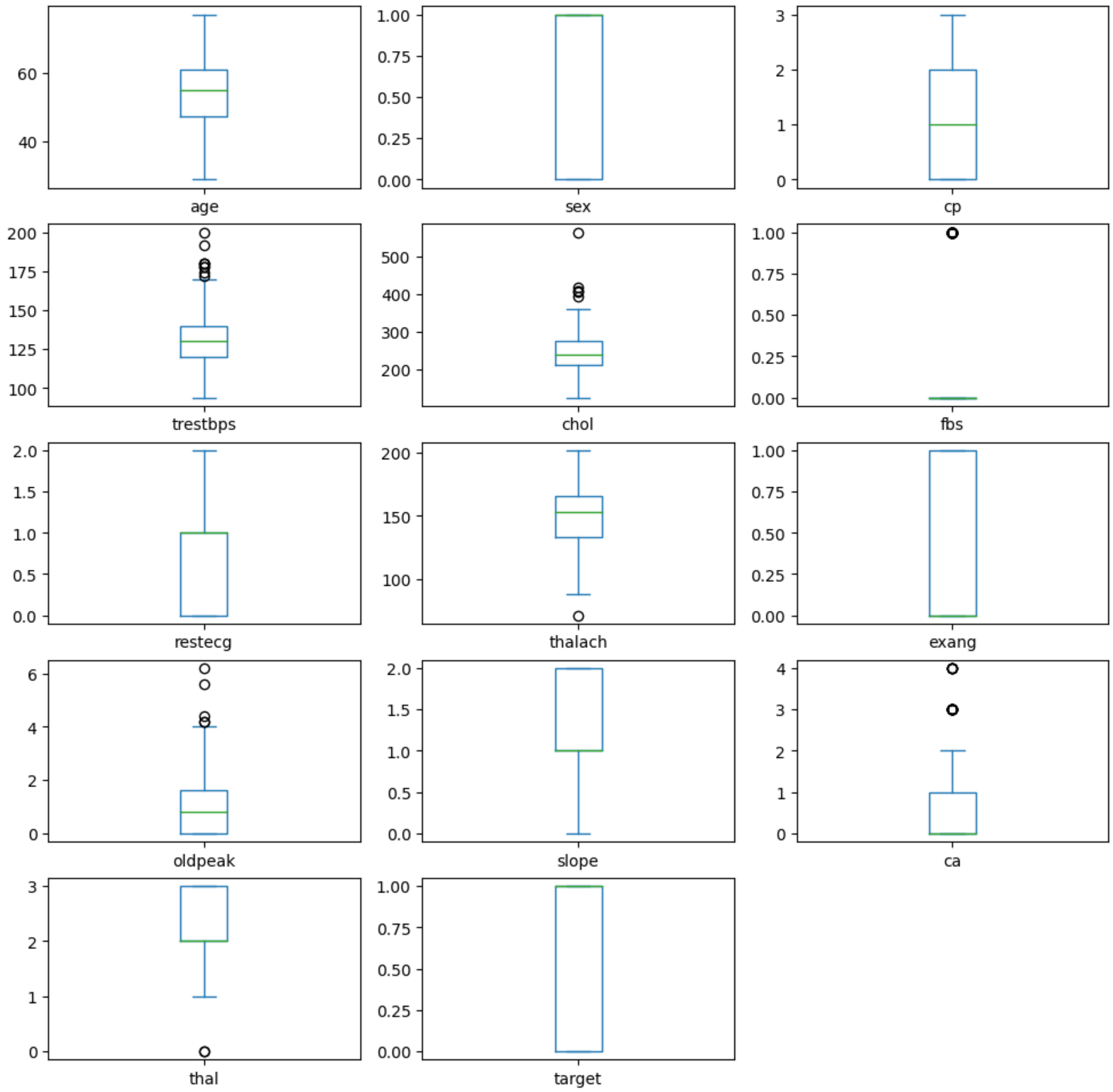



```
[12]: sns.pairplot(data, hue= 'target')
```

```
[12]: <seaborn.axisgrid.PairGrid at 0x115c2c29450>
```



[11]:
```
data.plot(kind='box',subplots=True,figsize=(12,12),layout=(5,3))
plt.show()
```

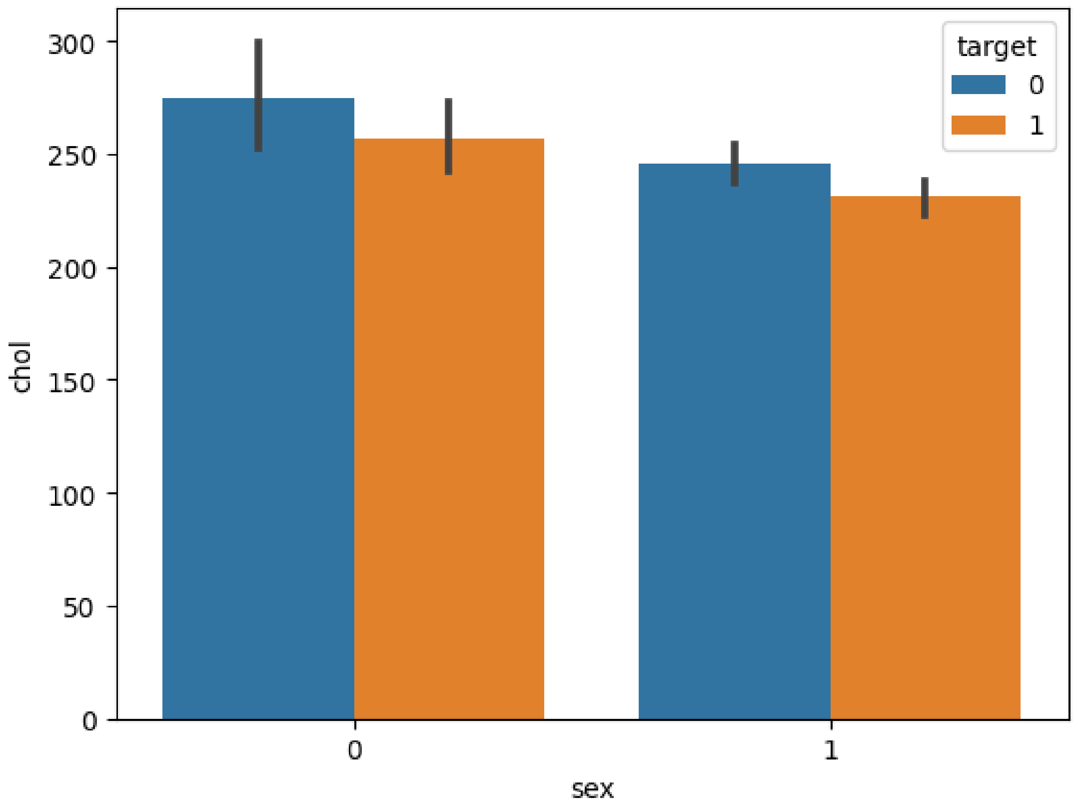


[313]:
```
sns.catplot(data=data, x='sex', y='age', hue='target',palette='Pastel2')
plt.title('Sex vs Age')
```

[313]: Text(0.5, 1.0, 'Sex vs Age')

[314]:
```
sns.barplot(data=data,x='sex',y='chol',hue='target')
```

[314]: <Axes: xlabel='sex', ylabel='chol'>



[315]:
```
data['sex'].value_counts()
```

[315]:
```
sex
1    207
0     96
Name: count, dtype: int64
```

[316]:
```
#207 males and 96 females
```

[317]:
```
data['target'].value_counts()
```

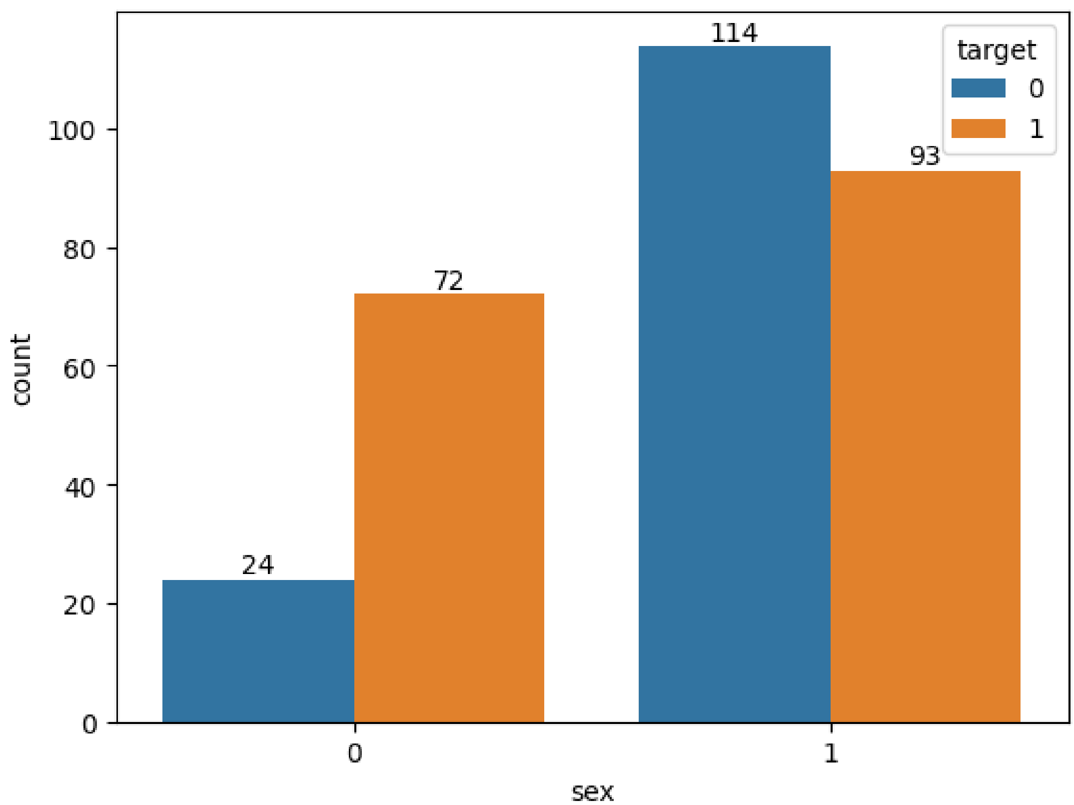
[317]:
```
target
1    165
0    138
Name: count, dtype: int64
```

[318]:
```
#165 have heart diseases out of 303
```

[319]:
```
data['thal'].value_counts()
```

[319]: thal
2   166
3   117
1   18
0   2
Name: count, dtype: int64

[320]:
```
x = sns.countplot(x='sex',data=data,hue='target')
for label in x.containers:
    x.bar_label(label)
```



0.1 Thal analysis

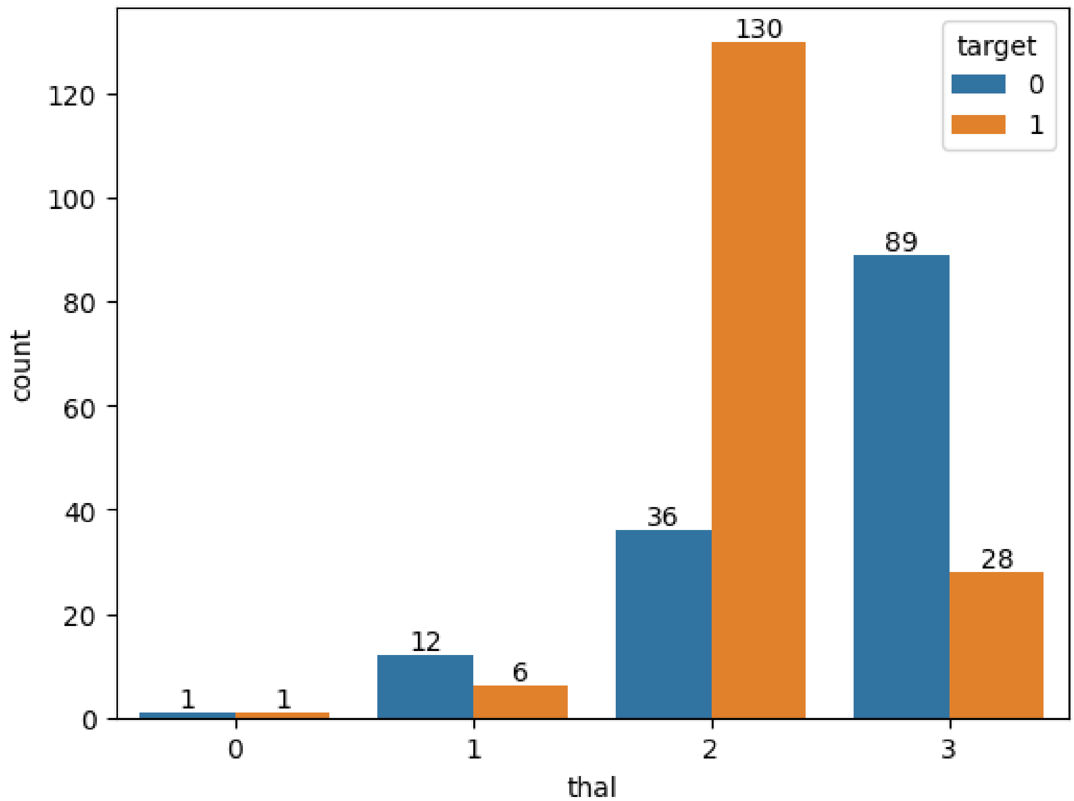
thal: A blood disorder called thalassemia Value 0: NULL (dropped from the dataset previously Value 1: fixed defect (no blood flow in some part of the heart) Value 2: normal blood flow Value 3: reversible defect (a blood flow is observed but it is not normal)

[321]:
```
data['thal'].value_counts()
```

[321]: thal
2   166

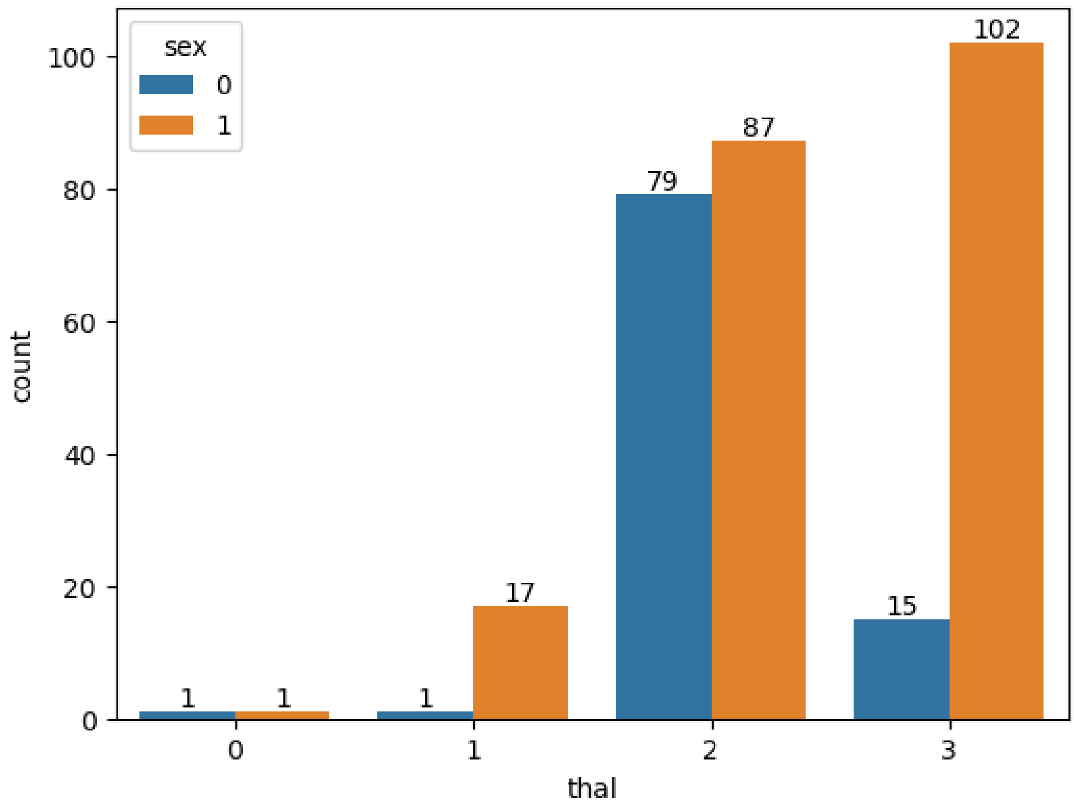
```
3    117
1     18
0      2
Name: count, dtype: int64
```

[322]:
```
a = sns.countplot(x='thal',data=data,hue='target')
for label in a.containers:
    a.bar_label(label)
```



[323]:
```
#thal value 2 has highest heart disease patients
```

[324]:
```
b = sns.countplot(x='thal',data=data,hue='sex')
for label in b.containers:
    b.bar_label(label)
```



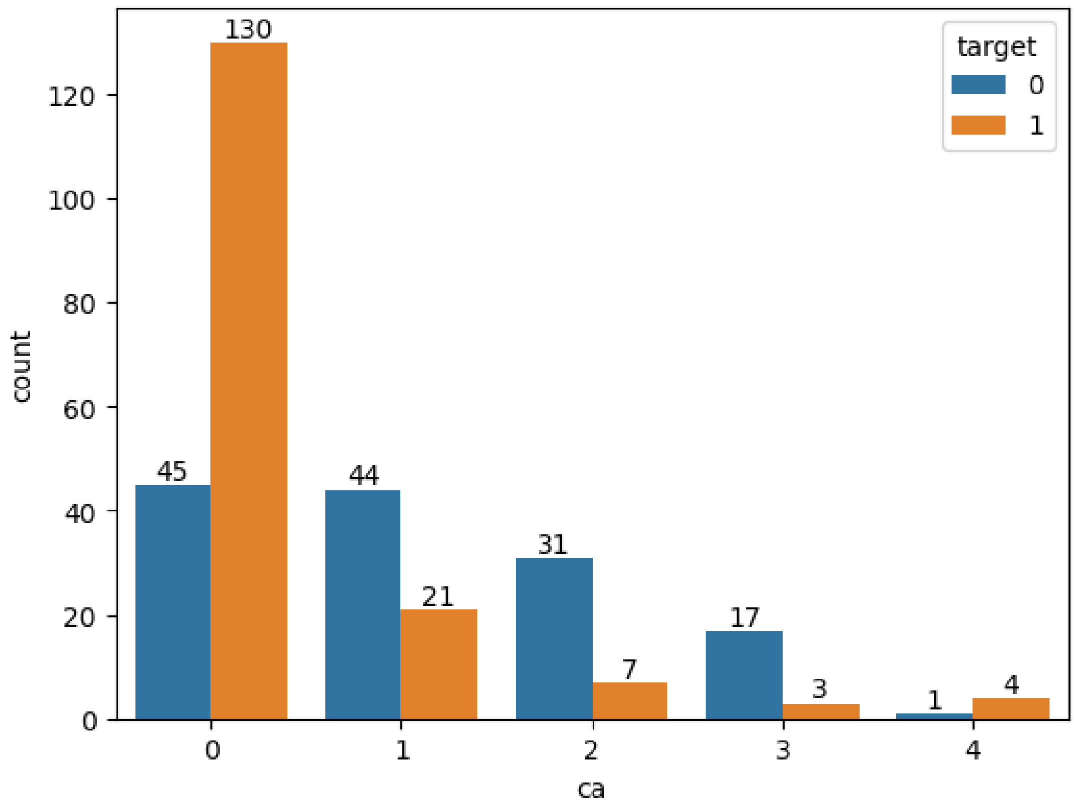
0.2 CA analysis

ca: The number of major vessels (0–3)

[325]:
```
data['ca'].value_counts()
```

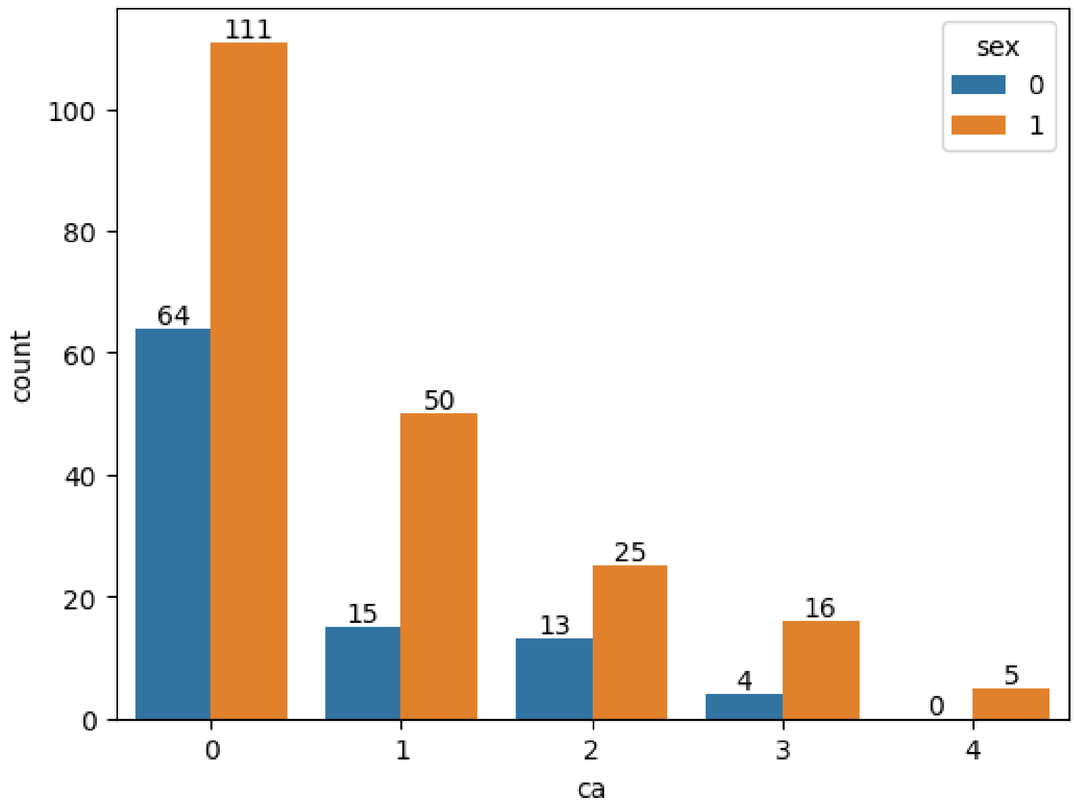
[325]:
```
ca
0    175
1     65
2     38
3     20
4      5
Name: count, dtype: int64
```

[326]:
```
c = sns.countplot(x='ca',data=data,hue='target')
for label in c.containers:
    c.bar_label(label)
```



[327]:
```
#ca value 0 has highest heart patients
```

[328]:
```
d = sns.countplot(x='ca',data=data,hue='sex')
for label in d.containers:
    d.bar_label(label)
```



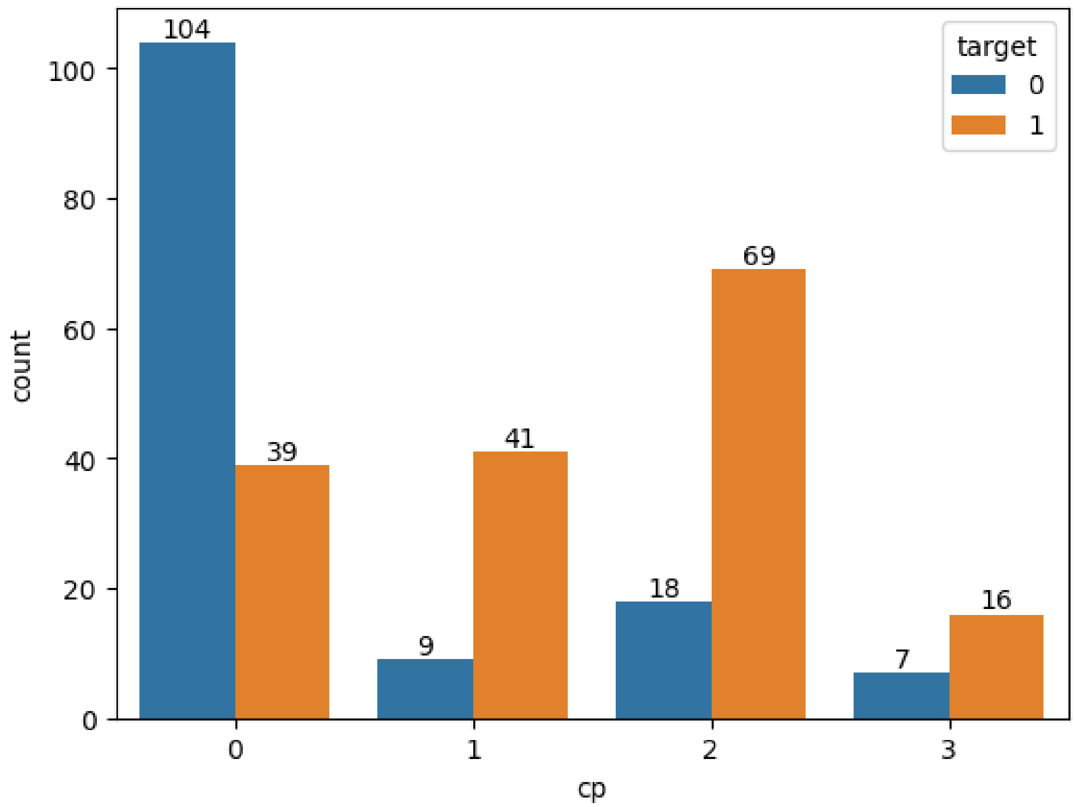
0.3 CP analysis

cp: chest pain type — Value 0: asymptomatic — Value 1: atypical angina — Value 2: non-anginal pain — Value 3: typical angina

[329]:
```
data['cp'].value_counts()
```

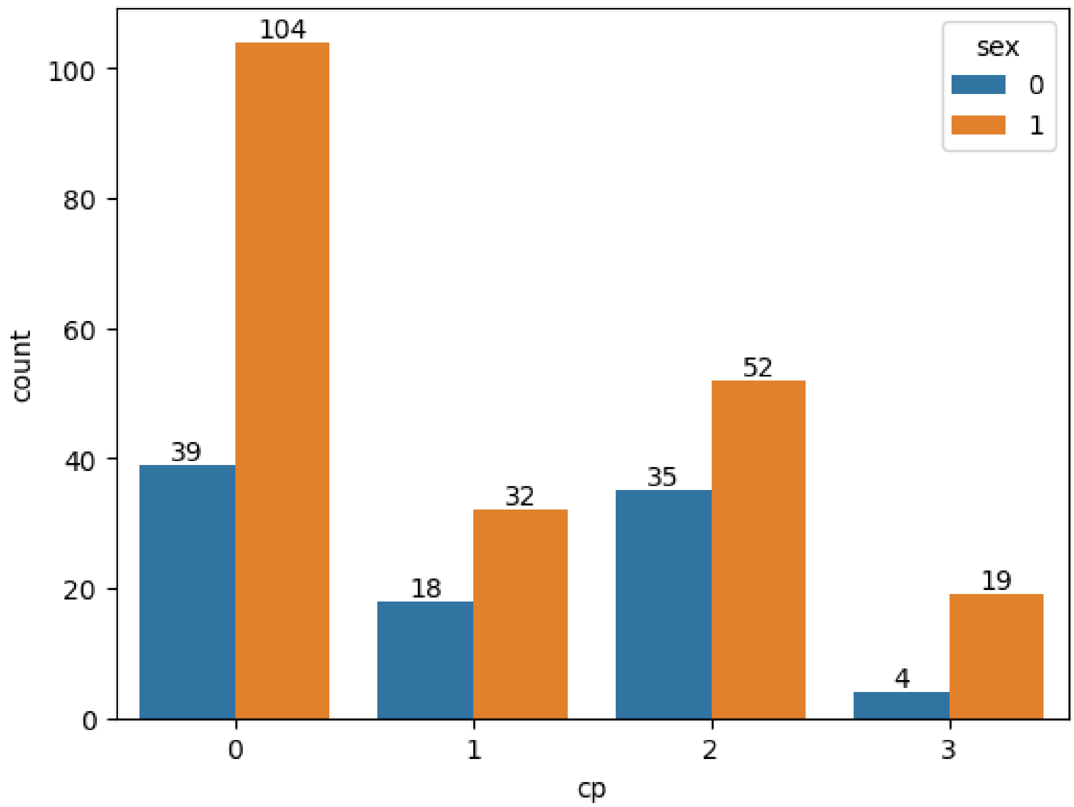
[329]:
```
cp
0    143
2     87
1     50
3     23
Name: count, dtype: int64
```

[330]:
```
e = sns.countplot(x='cp',data=data,hue='target')
for label in e.containers:
    e.bar_label(label)
```



[331]:
```
#cp value 2 has highest heart disease patients
```

[332]:
```
f = sns.countplot(x='cp',data=data,hue='sex')
for label in f.containers:
    f.bar_label(label)
```



0.4 Old peak analysis

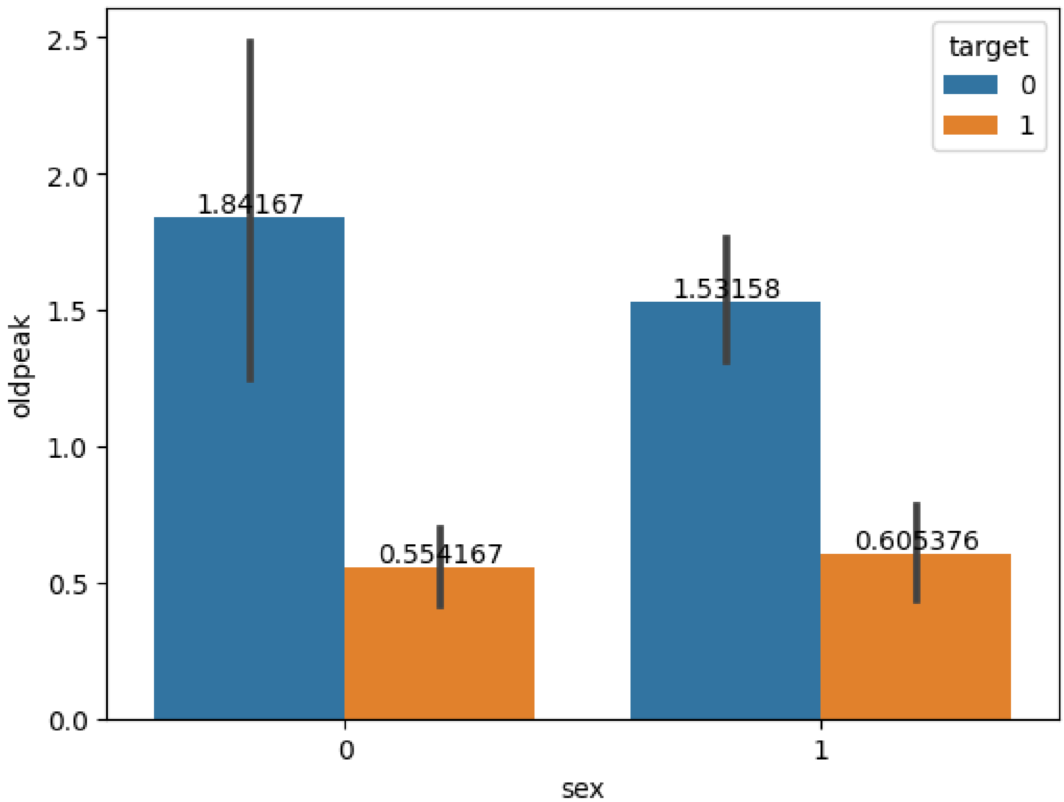
oldpeak: ST depression induced by exercise relative to rest ('ST' relates to positions on the ECG plot. See more here)

[333]:
```
data['oldpeak'].value_counts()
```

[333]:
```
oldpeak
0.0    99
1.2    17
1.0    14
0.6    14
1.4    13
0.8    13
0.2    12
1.6    11
1.8    10
0.4     9
2.0     9
0.1     7
2.8     6
```

```
2.6     6
1.5     5
3.0     5
1.9     5
0.5     5
3.6     4
2.2     4
2.4     3
0.9     3
3.4     3
4.0     3
0.3     3
2.3     2
3.2     2
2.5     2
4.2     2
1.1     2
3.1     1
0.7     1
3.5     1
6.2     1
1.3     1
5.6     1
2.9     1
2.1     1
3.8     1
4.4     1
Name: count, dtype: int64
```

[334]:
```
g = sns.barplot(y='oldpeak',x='sex',data=data,hue='target')
for label in g.containers:
    g.bar_label(label)
```



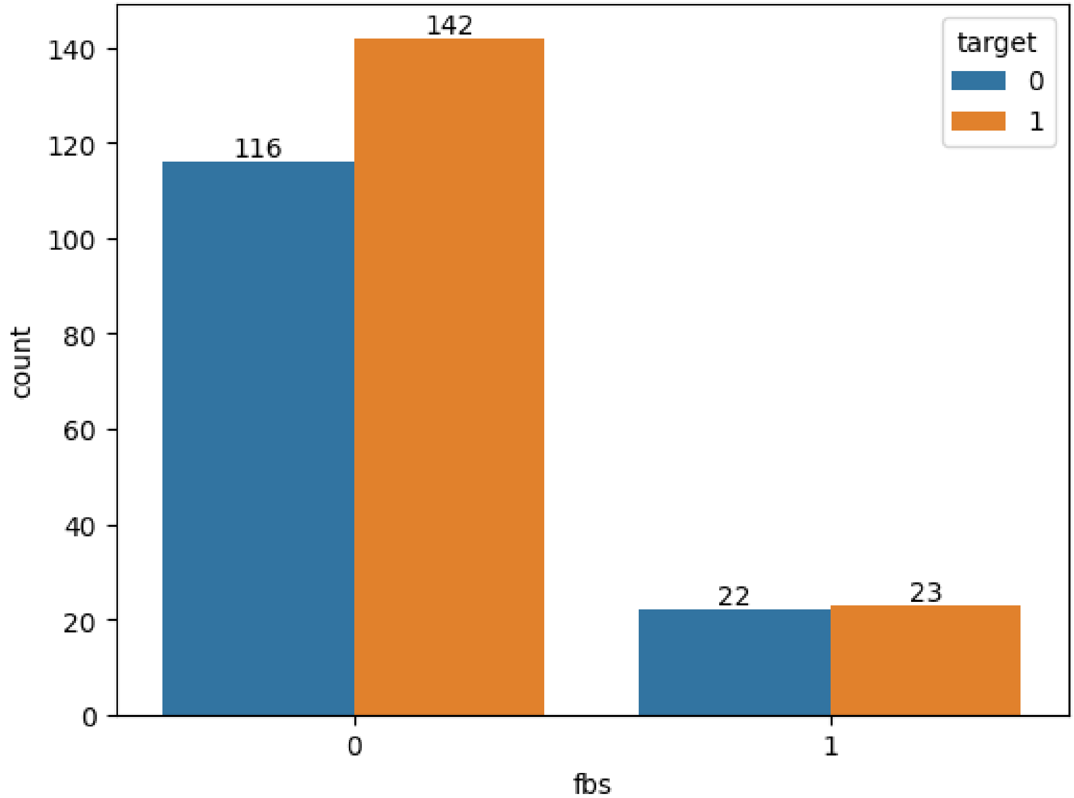
0.5 Fbs analysis

The person's fasting blood sugar (> 120 mg/dl, 1 = true; 0 = false)

[335]:
```
data['fbs'].value_counts()
```

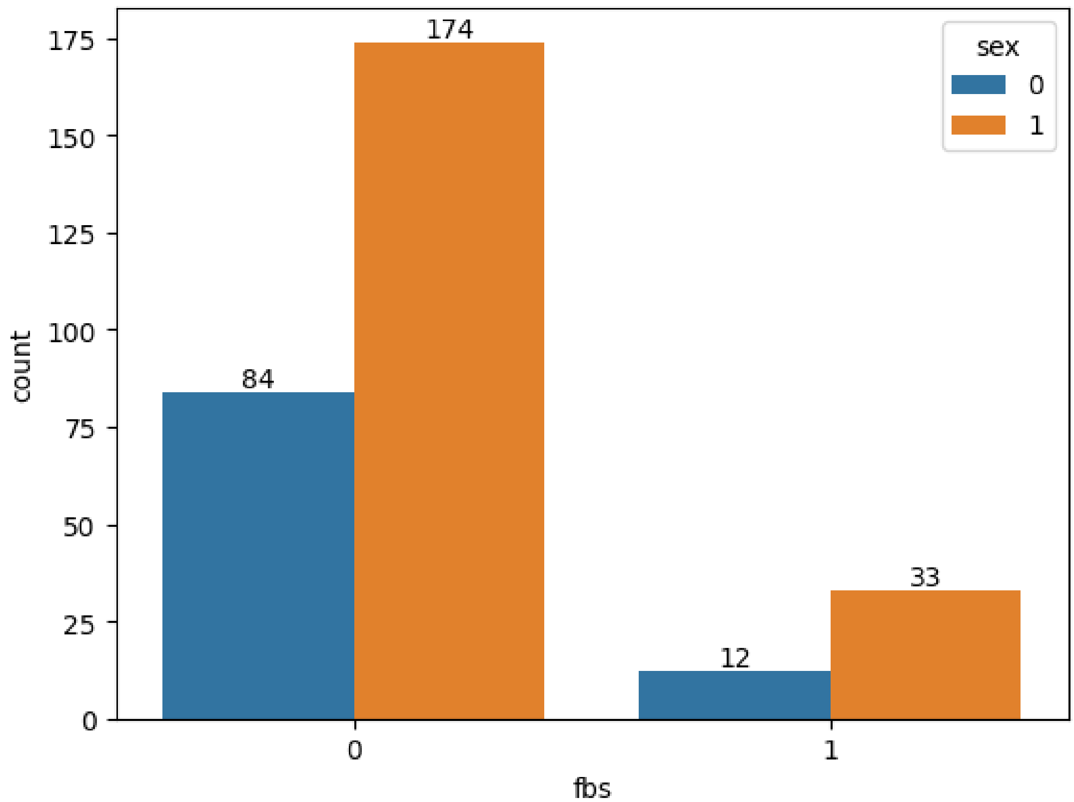
[335]:
```
fbs
0    258
1     45
Name: count, dtype: int64
```

[336]:
```
h = sns.countplot(x='fbs',data=data,hue='target')
for label in h.containers:
    h.bar_label(label)
```



[337]:
```
#if fbs is lower than 120 then chances of heart diseases are higher
```

[338]:
```
i = sns.countplot(x='fbs',data=data,hue='sex')
for label in i.containers:
    i.bar_label(label)
```



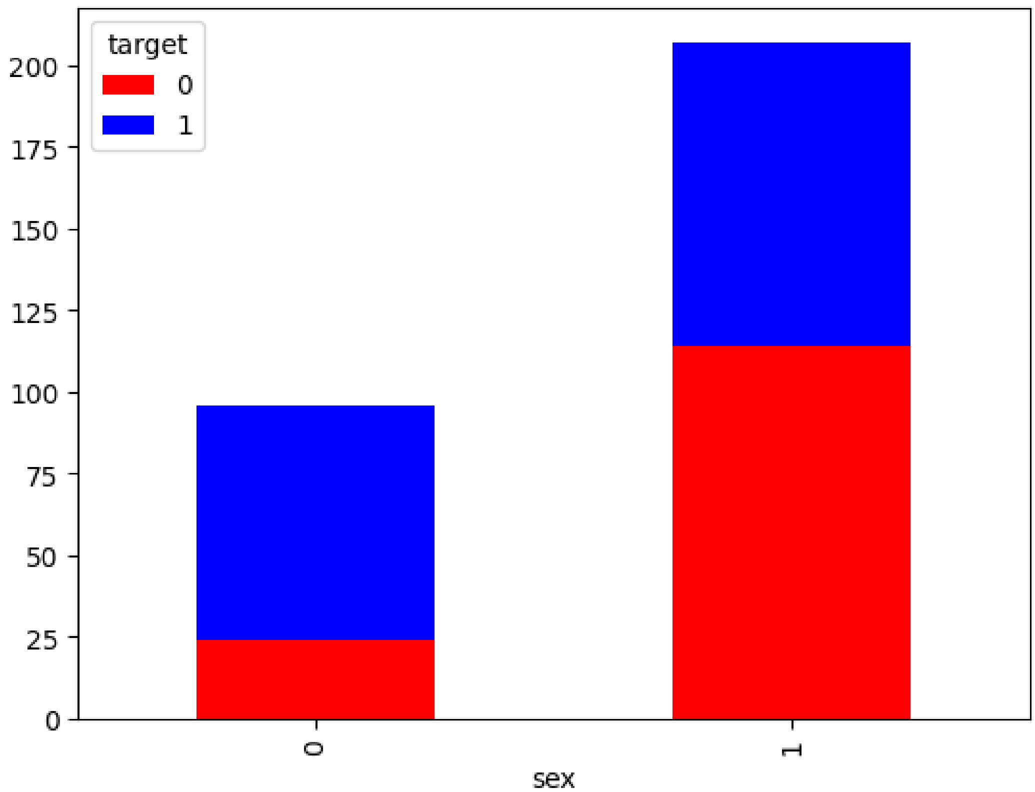
0.6 Cross Feature Analysis

[339]:
```
target_and_sex = pd.crosstab(data['sex'], data['target'])
print(target_and_sex)
```

```
target    0   1
sex
0        24  72
1       114  93
```

[340]:
```
target_and_sex.plot(kind='bar',stacked=True, color=['red','blue'])
```

[340]: <Axes: xlabel='sex'>

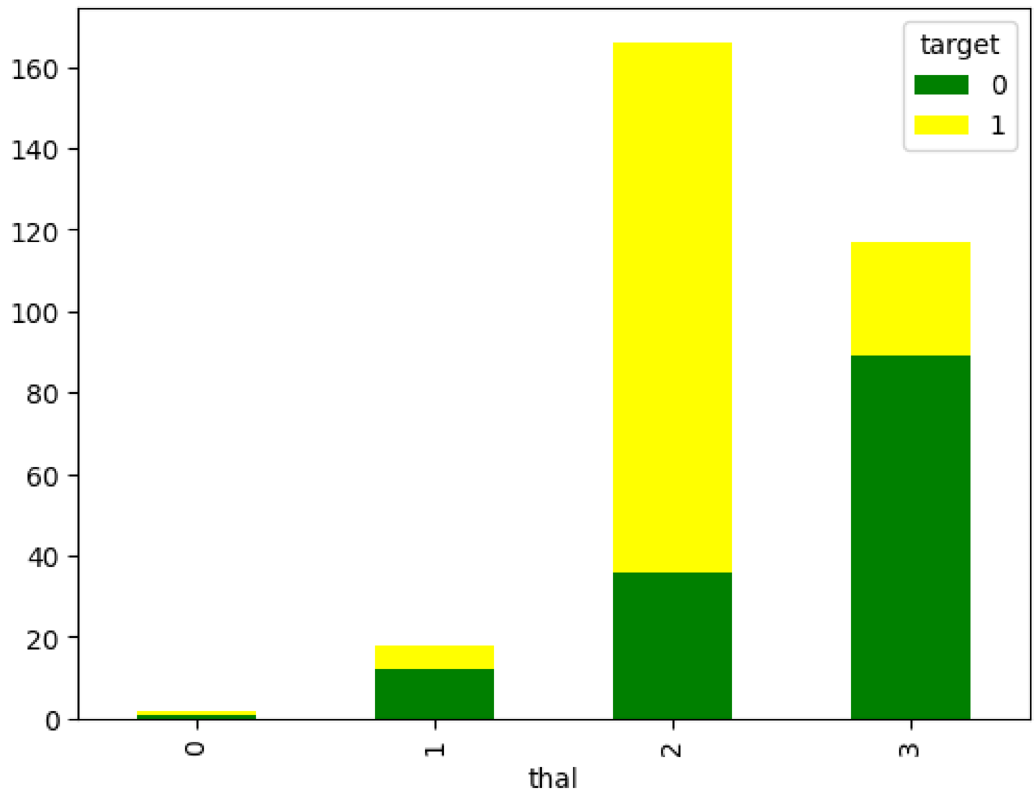


[341]:
```
thal_and_target = pd.crosstab(data['thal'], data['target'])
print(thal_and_target)
```

```
target   0    1
thal
0        1    1
1       12    6
2       36  130
3       89   28
```

[342]:
```
thal_and_target.plot(kind='bar',stacked=True, color=['green','yellow'])
```

[342]: <Axes: xlabel='thal'>

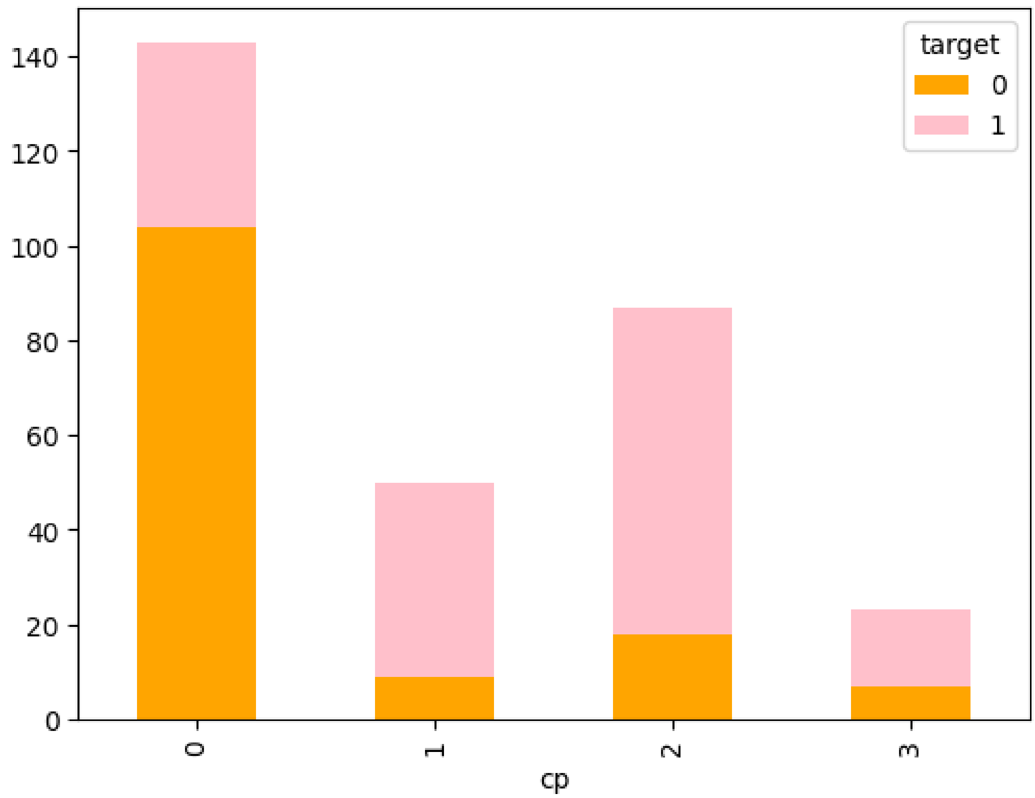


[343]:
```python
cp_and_target = pd.crosstab(data['cp'], data['target'])
print(cp_and_target)
```

```
target    0   1
cp
0       104  39
1         9  41
2        18  69
3         7  16
```

[344]:
```python
cp_and_target.plot(kind='bar',stacked=True, color=['orange','pink'])
```

[344]: <Axes: xlabel='cp'>

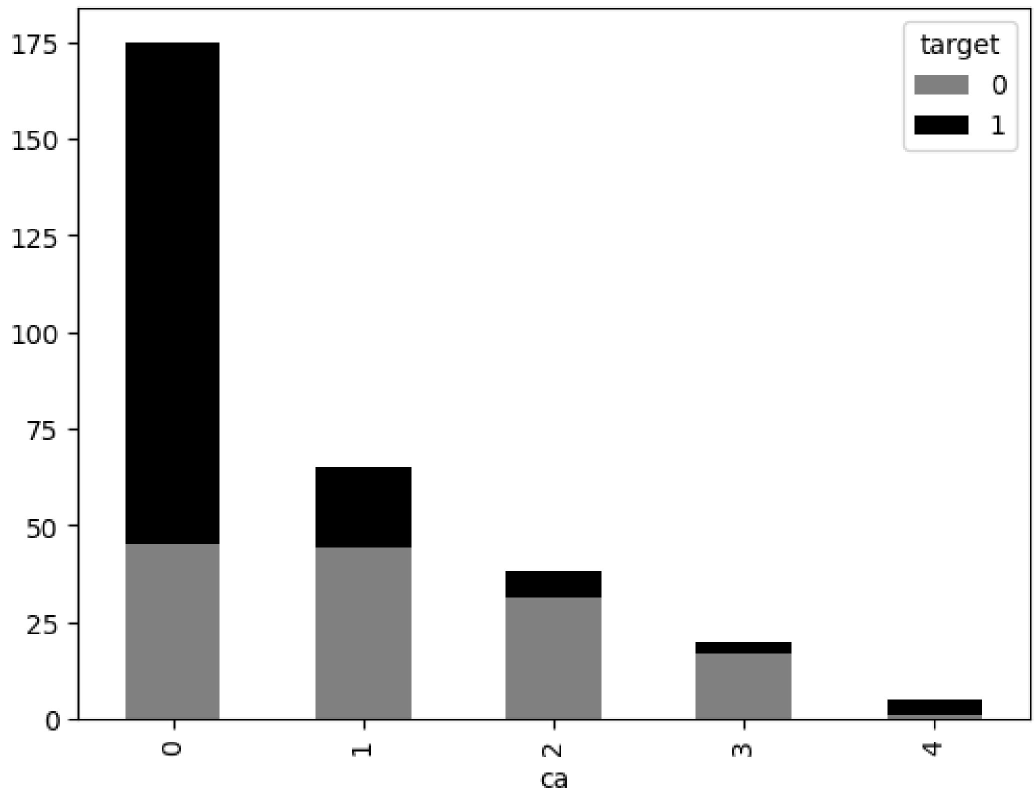


[345]:
```
ca_and_target = pd.crosstab(data['ca'], data['target'])
print(ca_and_target)
```

```
target   0    1
ca
0       45  130
1       44   21
2       31    7
3       17    3
4        1    4
```

[346]:
```
ca_and_target.plot(kind='bar',stacked=True, color=['grey','black'])
```

[346]: <Axes: xlabel='ca'>



0.7 Model Building

[347]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
StandardScaler = StandardScaler()
columns_to_scale = ['age','trestbps','chol','thalach','oldpeak']
data[columns_to_scale] = StandardScaler.fit_transform(data[columns_to_scale])
```

[348]:
```
data.head()
```

[348]:
```
        age  sex  cp  trestbps      chol  fbs  restecg   thalach  exang  \
0  0.952197    1   3  0.763956 -0.256334    1        0  0.015443      0
1 -1.915313    1   2 -0.092738  0.072199    0        1  1.633471      0
2 -1.474158    0   1 -0.092738 -0.816773    0        0  0.977514      0
3  0.180175    1   1 -0.663867 -0.198357    0        1  1.239897      0
4  0.290464    0   0 -0.663867  2.082050    0        1  0.583939      1

    oldpeak  slope  ca  thal  target
0  1.087338      0   0     1       1
1  2.122573      0   0     2       1
```

```
2  0.310912       2   0     2        1
3 -0.206705       2   0     2        1
4 -0.379244       2   0     2        1
```

[354]:
```
X = data.drop(['target'], axis=1)
Y = data['target']
```

[355]:
```
X
```

[355]:
```
          age  sex  cp  trestbps      chol  fbs  restecg   thalach  exang  
0    0.952197    1   3  0.763956 -0.256334    1        0  0.015443      0   \
1   -1.915313    1   2 -0.092738  0.072199    0        1  1.633471      0   
2   -1.474158    0   1 -0.092738 -0.816773    0        0  0.977514      0   
3    0.180175    1   1 -0.663867 -0.198357    0        1  1.239897      0   
4    0.290464    0   0 -0.663867  2.082050    0        1  0.583939      1   
..        …    …  ..       …         …  …        …       …      
298  0.290464    0   0  0.478391 -0.101730    0        1 -1.165281      1   
299 -1.033002    1   3 -1.234996  0.342756    0        1 -0.771706      0   
300  1.503641    1   0  0.706843 -1.029353    1        1 -0.378132      0   
301  0.290464    1   0 -0.092738 -2.227533    0        1 -1.515125      1   
302  0.290464    0   1 -0.092738 -0.198357    0        0  1.064975      0   

      oldpeak  slope  ca  thal
0    1.087338      0   0     1
1    2.122573      0   0     2
2    0.310912      2   0     2
3   -0.206705      2   0     2
4   -0.379244      2   0     2
..        …    …  ..   …
298 -0.724323      1   0     3
299  0.138373      1   0     3
300  2.036303      1   2     3
301  0.138373      1   1     3
302 -0.896862      1   1     2

[303 rows x 13 columns]
```

[356]:
```
Y
```

[356]:
```
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
```

[357]:
```
X_train, X_test,y_train, y_test= train_test_split(X,Y,test_size=0.
 ↪3,random_state=40)
```

[358]:
```
print('X_train:', X_train.size)
print('X_test:',X_test.size)
print('y_train:', y_train.size)
print('y_test:', y_test.size)
```

```
X_train: 2756
X_test: 1183
y_train: 212
y_test: 91
```

0.8 Logistic Regression

[359]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
model1 = lr.fit(X_train,y_train)
prediction1 = model1.predict(X_test)
```

[360]:
```
prediction1
```

[360]:
```
array([0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 1], dtype=int64)
```

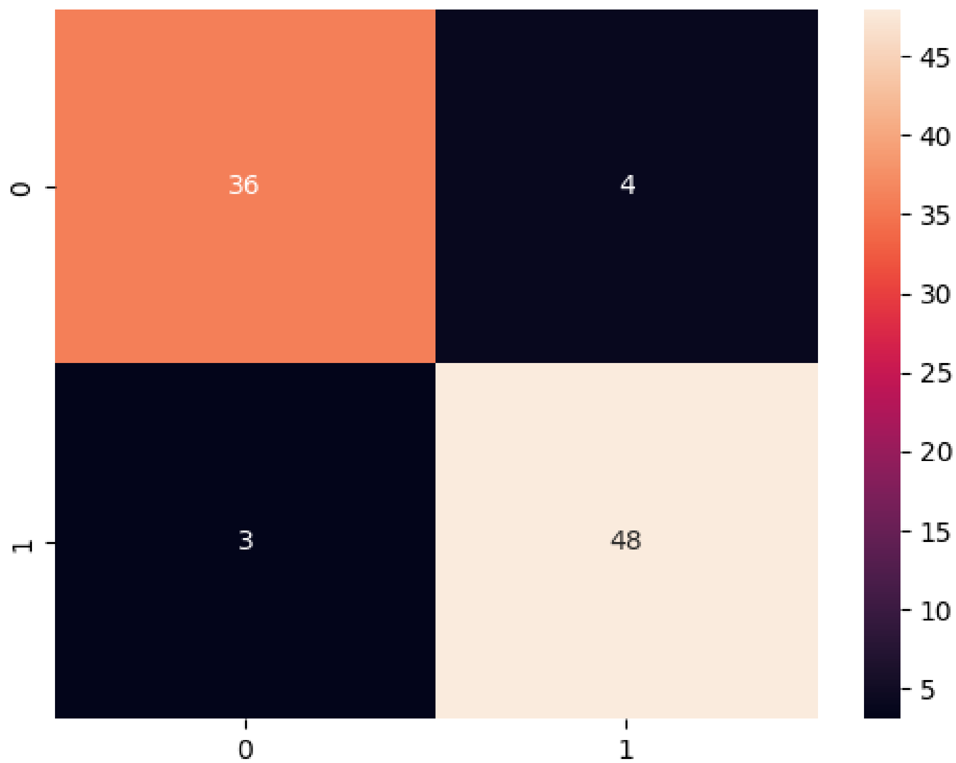
[361]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix1 = confusion_matrix(y_test,prediction1)
```

[362]:
```
confusion_matrix1
```

[362]:
```
array([[36,  4],
       [ 3, 48]], dtype=int64)
```

[363]:
```
sns.heatmap(confusion_matrix1, annot=True)
```

[363]:
```
<Axes: >
```



[364]:
```
TP=confusion_matrix1[0][0]
TN=confusion_matrix1[1][1]
FN=confusion_matrix1[1][0]
FP=confusion_matrix1[0][1]
print('Testing Accuracy:',(TP+TN)/(TP+TN+FN+FP))
```

```
Testing Accuracy: 0.9230769230769231
```

0.9 Random Forest

[365]:
```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
model2 = rfc.fit(X_train, y_train)
prediction2 = model2.predict(X_test)
```

[366]:
```
prediction2
```

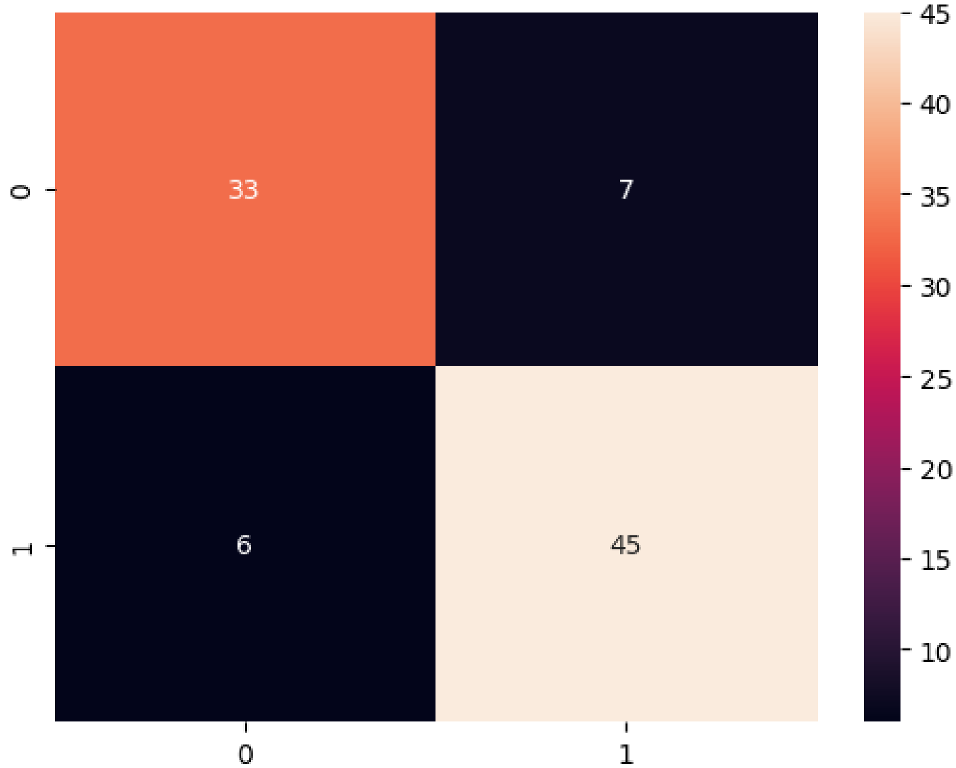
[366]:
```
array([1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1,
```

```
        1, 1, 1], dtype=int64)
```

[367]:
```
confusion_matrix2 = confusion_matrix(y_test, prediction2)
```

[368]:
```
sns.heatmap(confusion_matrix2, annot=True)
```

[368]: <Axes: >



[369]:
```
TP=confusion_matrix2[0][0]
TN=confusion_matrix2[1][1]
FN=confusion_matrix2[1][0]
FP=confusion_matrix2[0][1]
print('Testing Accuracy:',(TP+TN)/(TP+TN+FN+FP))
```

```
Testing Accuracy: 0.8571428571428571
```

0.10 Naive Bayes

[370]:
```
from sklearn.naive_bayes import GaussianNB
NB = GaussianNB()
model3 = NB.fit(X_train, y_train)
prediction3 = model3.predict(X_test)
```

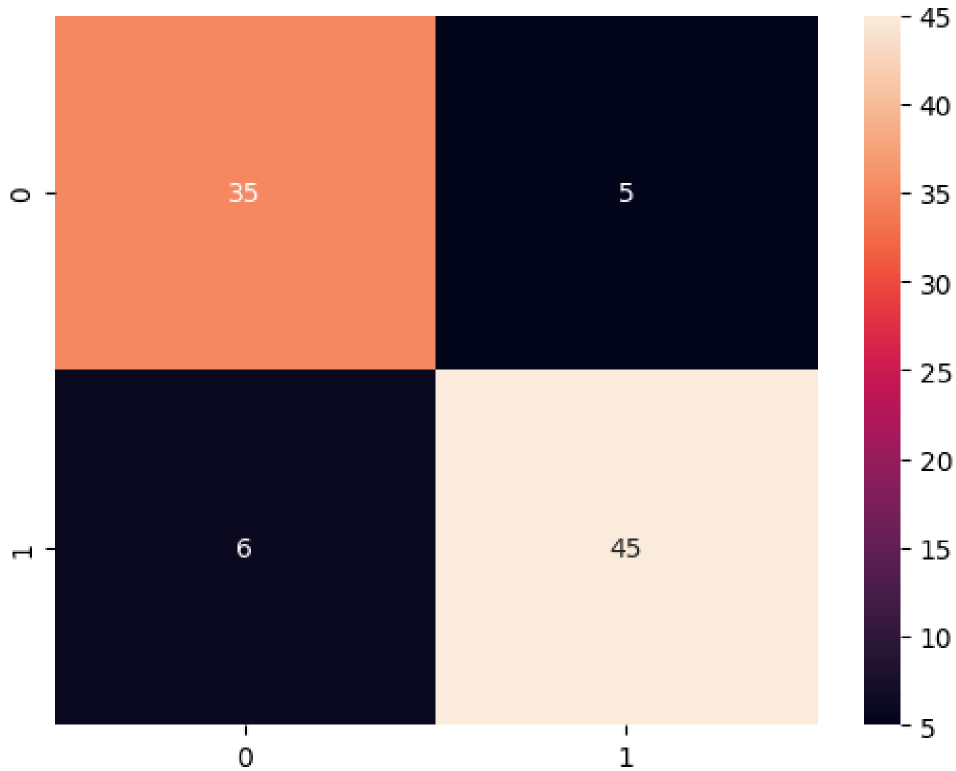
[371]:
```
prediction3
```

[371]:
```
array([1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 1], dtype=int64)
```

[372]:
```
confusion_matrix3 = confusion_matrix(y_test, prediction3)
```

[373]:
```
sns.heatmap(confusion_matrix3, annot=True)
```

[373]: <Axes: >



[374]:
```
TP=confusion_matrix3[0][0]
TN=confusion_matrix3[1][1]
FN=confusion_matrix3[1][0]
FP=confusion_matrix3[0][1]
print('Testing Accuracy:',(TP+TN)/(TP+TN+FN+FP))
```

```
Testing Accuracy: 0.8791208791208791
```

0.11 KNN

[375]:
```
from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier()
model4 = KNN.fit(X_train, y_train)
prediction4 = model4.predict(X_test)
```

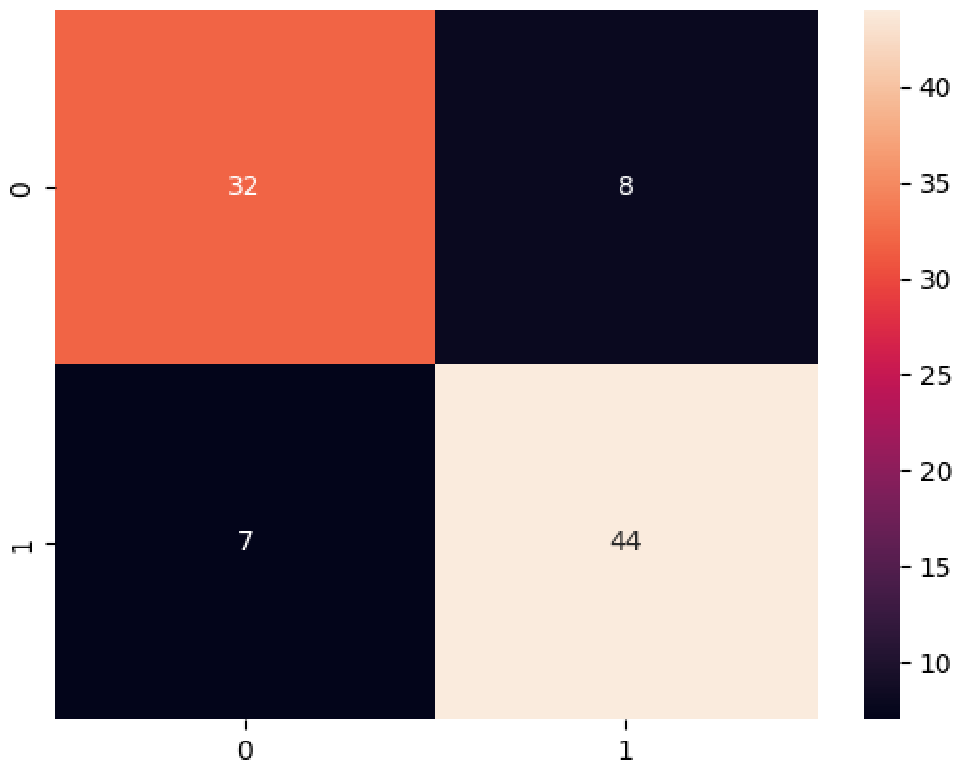
[376]:
```
prediction4
```

[376]:
```
array([1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1,
       0, 0, 1], dtype=int64)
```

[377]:
```
confusion_matrix4 = confusion_matrix(y_test, prediction4)
```

[378]:
```
sns.heatmap(confusion_matrix4, annot=True)
```

[378]: <Axes: >



[379]:
```
TP=confusion_matrix4[0][0]
TN=confusion_matrix4[1][1]
FN=confusion_matrix4[1][0]
FP=confusion_matrix4[0][1]
print('Testing Accuracy:',(TP+TN)/(TP+TN+FN+FP))
```

```
Testing Accuracy: 0.8351648351648352
```

0.12 Decision Tree

[380]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
model5 = dtc.fit(X_train,y_train)
prediction5 = model5.predict(X_test)
```

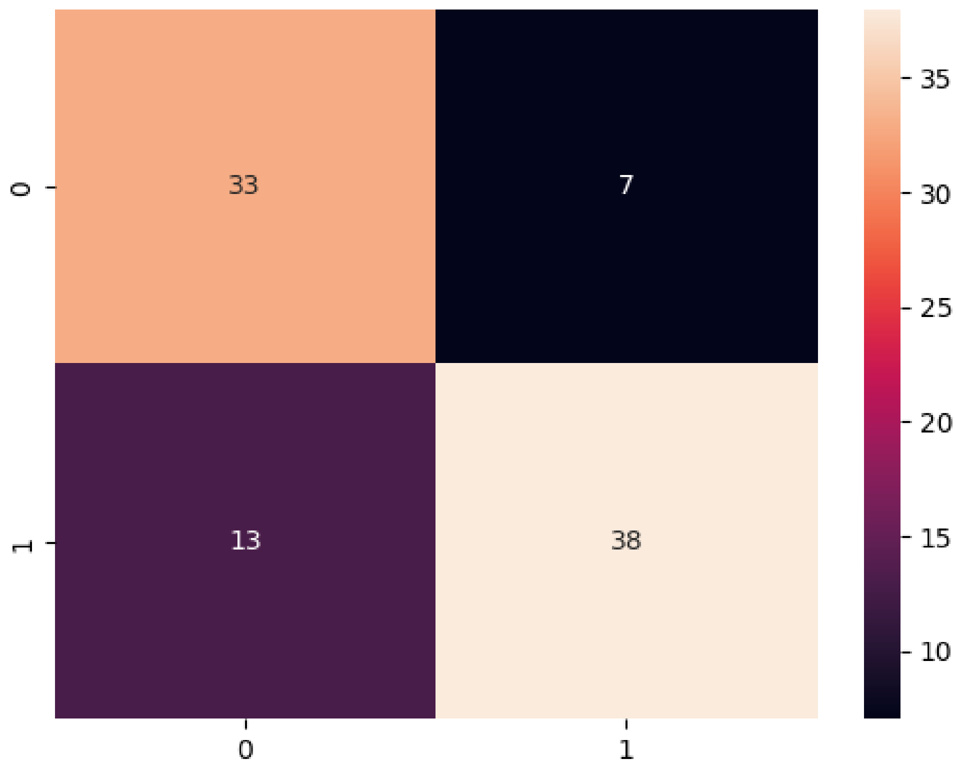
[381]:
```
prediction5
```

[381]:
```
array([1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0,
       1, 0, 1], dtype=int64)
```

[382]:
```
confusion_matrix5 = confusion_matrix(y_test, prediction5)
```

[383]:
```
sns.heatmap(confusion_matrix5, annot=True)
```

[383]:
```
<Axes: >
```



[384]:
```
TP=confusion_matrix5[0][0]
TN=confusion_matrix5[1][1]
FN=confusion_matrix5[1][0]
FP=confusion_matrix5[0][1]
print('Testing Accuracy:',(TP+TN)/(TP+TN+FN+FP))
```

```
Testing Accuracy: 0.7802197802197802
```

0.13 SVM

[385]:
```
from sklearn.svm import SVC
svm = SVC()
model6 = svm.fit(X_train,y_train)
prediction6 = model6.predict(X_test)
```

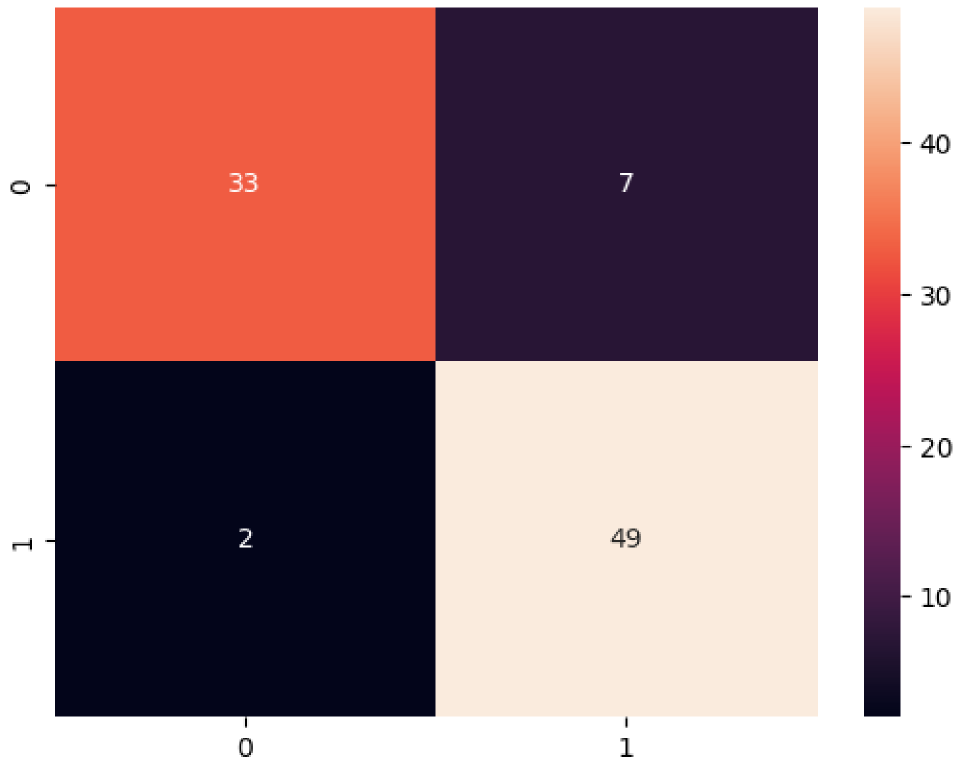
[386]:
```
prediction6
```

[386]:
```
array([1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 1], dtype=int64)
```

[387]:
```
confusion_matrix6 = confusion_matrix(y_test, prediction6)
```

[388]:
```
sns.heatmap(confusion_matrix6, annot=True)
```

[388]: <Axes: >



[389]:
```
TP=confusion_matrix6[0][0]
TN=confusion_matrix6[1][1]
FN=confusion_matrix6[1][0]
FP=confusion_matrix6[0][1]
print('Testing Accuracy:',(TP+TN)/(TP+TN+FN+FP))
```

```
Testing Accuracy: 0.9010989010989011
```

0.14 Comparing accuracies

[390]:
```
print('The accuracy of Logistic Regression :', accuracy_score(y_test,␣
  ↪prediction1))
print('The accuracy of Random Forest :', accuracy_score(y_test, prediction2))
print('The accuracy of Naive Bayes :', accuracy_score(y_test, prediction3))
print('The accuracy of KNN :', accuracy_score(y_test, prediction4))
print('The accuracy of Decision Tree :', accuracy_score(y_test, prediction5))
```

```python
print('The accuracy of SVM :', accuracy_score(y_test, prediction6))
```

```
The accuracy of Logistic Regression : 0.9230769230769231
The accuracy of Random Forest : 0.8571428571428571
The accuracy of Naive Bayes : 0.8791208791208791
The accuracy of KNN : 0.8351648351648352
The accuracy of Decision Tree : 0.7802197802197802
The accuracy of SVM : 0.9010989010989011
```

0.15 The best accuracy is achieved by Logistic Regression i.e 92.3%

[ ]:
```python

```