healthcare-data

June 23, 2024

Attribute Information:

Here we have the description of the features:

age - age in years

sex - sex (0 = female; 1 = male)

cp - chest pain type (1 = typical angina; 2 = atypical angina; 3 = non-anginal pain; 0 = asymptomatic)

trtbps - resting blood pressure (in mm Hg on admission to the hospital)

chol - serum cholestoral in mg/dl

fbs - fasting blood sugar > 120 mg/dl (0 = false; 1 = true)

restecg - resting electrocardiographic results (0 = normal; 1 = hypertrophy; 2 = having ST-T wave abnormality)

thalachh - maximum heart rate achieved

exng - exercise induced angina (0 = no; 1 = yes)

oldpeak - ST depression induced by exercise relative to rest

slp - the slope of the peak exercise ST segment (0 = downsloping; 1 = flat; 2 = upsloping)

caa - number of major vessels (0-4) colored by flourosopy

thall - thallium stress test (1 = fixed defect; 2 = reversable defect; 3 = normal)

output - 0 = less chance of heart attack; 1 = more chance of heart attack

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
data=pd.read_csv("B://Anaconda3//envs//nenvs//Heart Disease data//Heart Disease␣
↪data//Heart Disease data.csv")
```

[3]:
```
data.head()
```

[3]:
```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   52    1   0       125   212    0        1      168      0      1.0      2
1   53    1   0       140   203    1        0      155      1      3.1      0
2   70    1   0       145   174    0        1      125      1      2.6      0
3   61    1   0       148   203    0        1      161      0      0.0      2
4   62    0   0       138   294    1        1      106      0      1.9      1

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

[4]:
```python
data.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
```

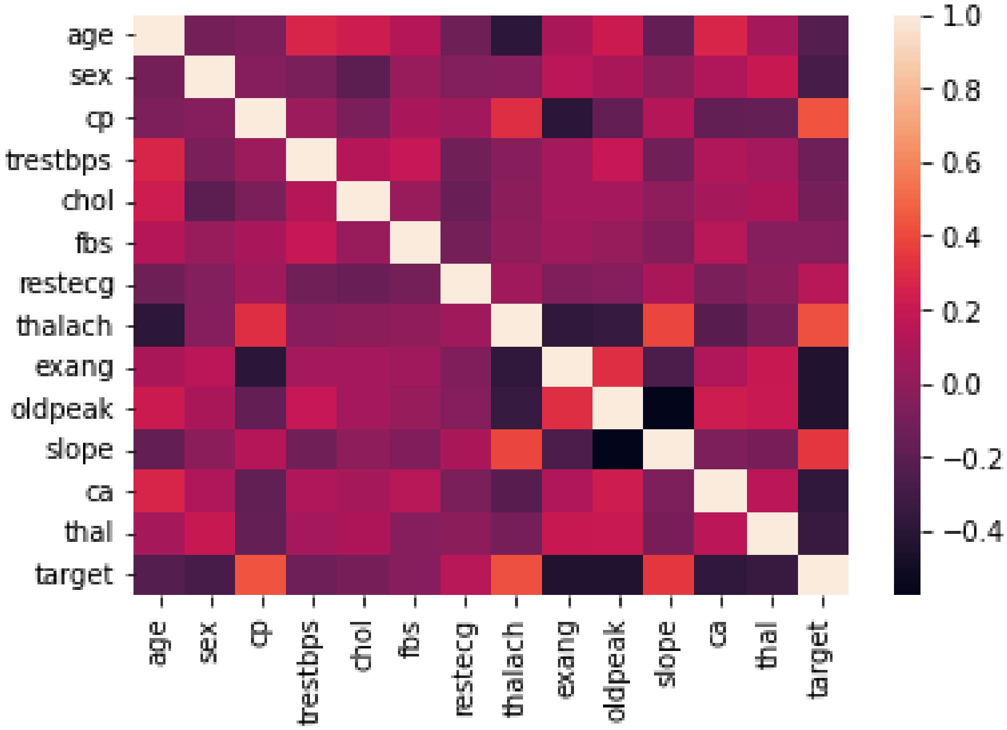
[5]:
```python
data.isnull().sum()
```

[5]:
```
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
```

[6]:
```
sns.heatmap(data.corr())
```

[6]:
```
<AxesSubplot:>
```



[7]:
```
#data['target']=data['target'].replace({0:'no chance of attack',1:'yes chance␣
↪of attack'})
```

[8]:
```
#data['sex']=data['sex'].replace({0:'Female',1:'Male'})
```

[9]:
```
'''
data['cp']=data['cp'].replace({1:'typical angina',2:'atypical angina',3:
↪'non-anginal pain',0:'asymptomatic'})
data['exang']=data['exang'].replace({0:'no',1:'yes'})
```

```
data['thal']=data['thal'].replace({1:'fixed defect',2:'reversible defect',3:
 ↪'normal'})
data['fbs']=data['fbs'].replace({0:'false',1:'true'})
'''
```

[9]:
```
"\ndata['cp']=data['cp'].replace({1:'typical angina',2:'atypical angina',3:'non-
anginal pain',0:'asymptomatic'})\ndata['exang']=data['exang'].replace({0:'no',1:
'yes'})\ndata['thal']=data['thal'].replace({1:'fixed defect',2:'reversible
defect',3:'normal'})\ndata['fbs']=data['fbs'].replace({0:'false',1:'true'})\n"
```

[10]:
```
#data['restecg']=data['restecg'].replace({0:'normal',1:'hypertrophy',2:'having␣
 ↪ST-T wave abnormality'})
#data['slope']=data['slope'].replace({0:'downsloping',1:'flat',2:'upsloping'})␣
 ↪*/
```

[11]:
```
#data.to_csv('HealthCare.csv')
```

Finding duplicates and removing duplicate values

[12]:
```
data[data.duplicated()]
```

[12]:
```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
15     34    0   1       118   210    0        1      192      0      0.7
31     50    0   1       120   244    0        1      162      0      1.1
43     46    1   0       120   249    0        0      144      0      0.8
55     55    1   0       140   217    0        1      111      1      5.6
61     66    0   2       146   278    0        0      152      0      0.0
…     …    …   ..        …     …    …        …       …      …        …
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
15        2   0     2       1
31        2   0     2       1
43        2   0     3       0
55        0   0     3       0
61        1   1     2       1
…       …  ..    …       …
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
```

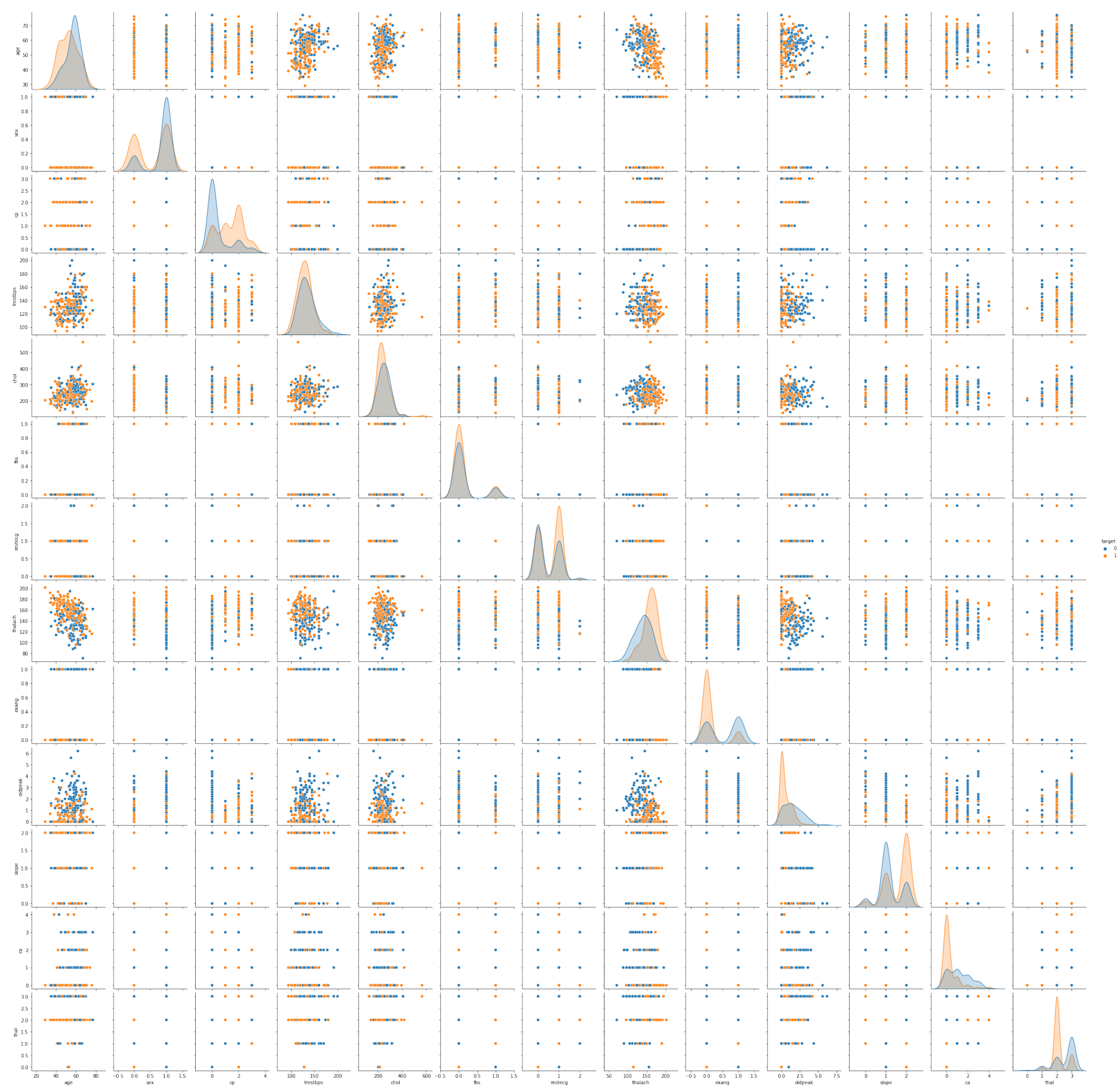
[723 rows x 14 columns]

[13]:
```
data.drop_duplicates(inplace=True)
data.shape
```

[13]: (302, 14)

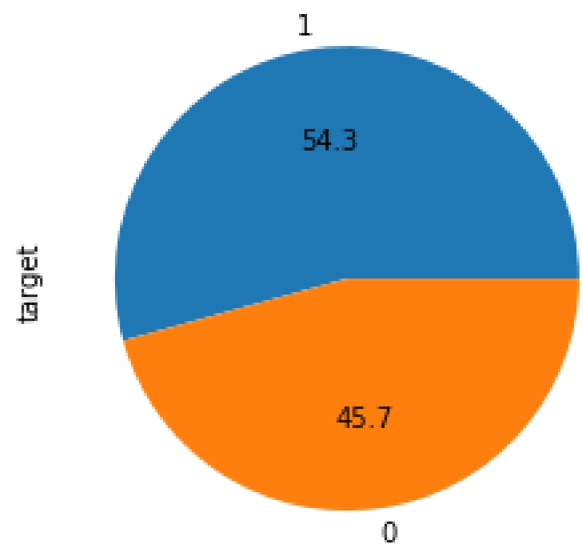
[14]:
```
sns.pairplot(data,hue='target')
```

[14]: <seaborn.axisgrid.PairGrid at 0x1eaeb20efd0>



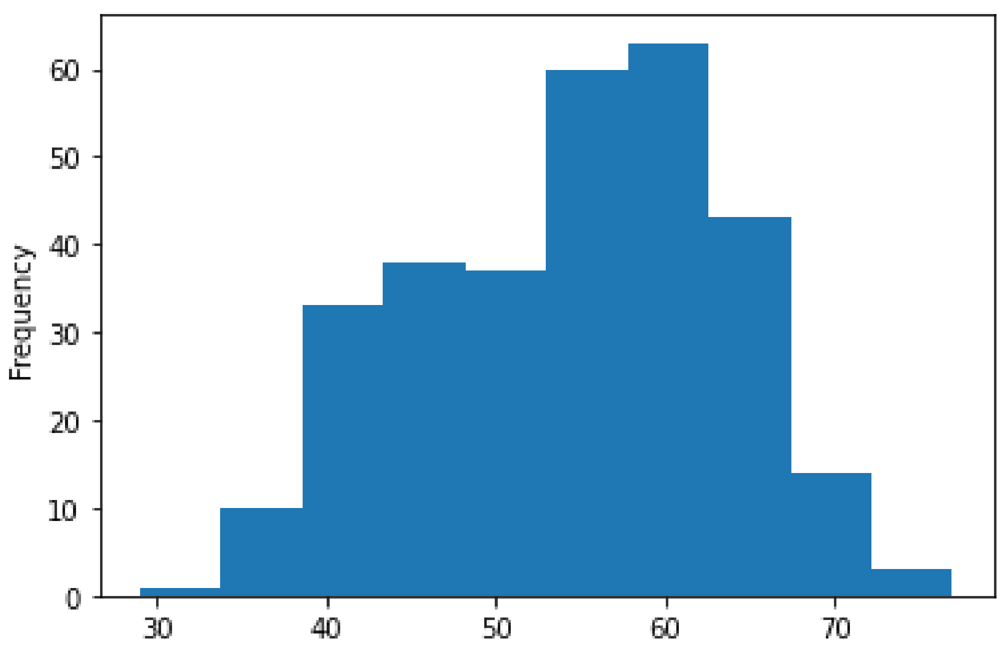
[15]:
```
data['target'].value_counts().plot.pie(autopct="%1.1f")
```

[15]: <AxesSubplot:ylabel='target'>



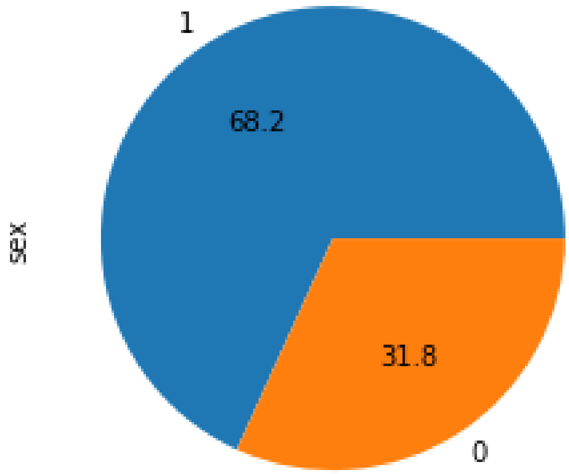

```
[16]: data['age'].plot.hist()
```

```
[16]: <AxesSubplot:ylabel='Frequency'>
```




```
[17]: data['sex'].value_counts().plot.pie(autopct="%1.1f")
```

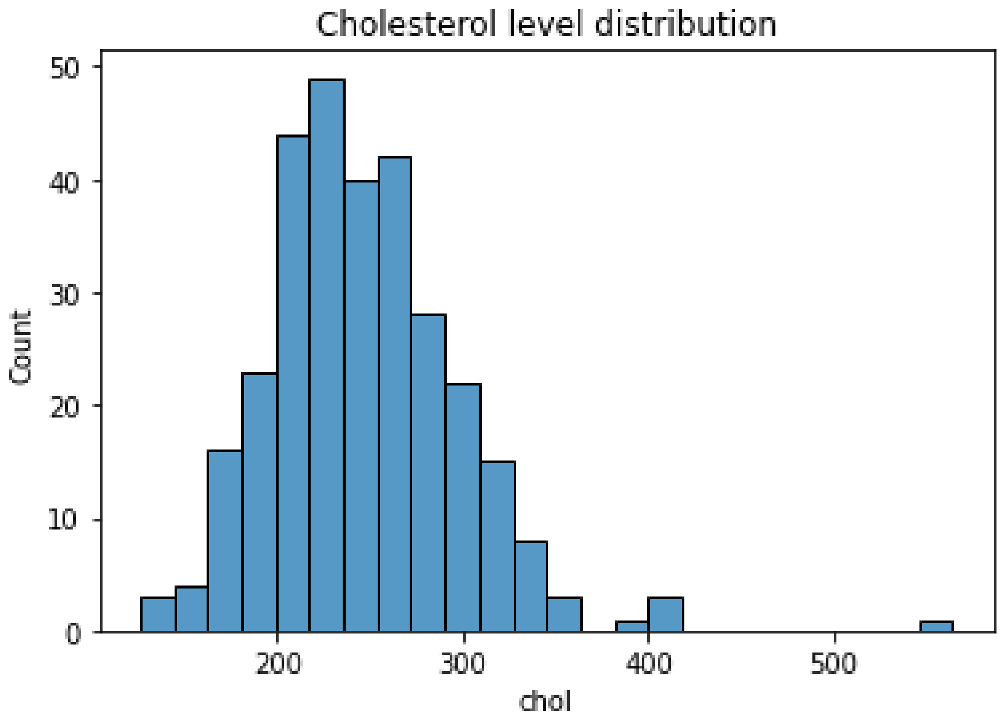
[17]: <AxesSubplot:ylabel='sex'>



Conclusion:

```
Male are double times likely to face heart problems than female
57-60+ range of age people are likely to face heart problems
57.5% of population are likely to have heart problems
```

[18]:
```
sns.histplot(data.chol)
plt.title('Cholesterol level distribution')
plt.show()
```



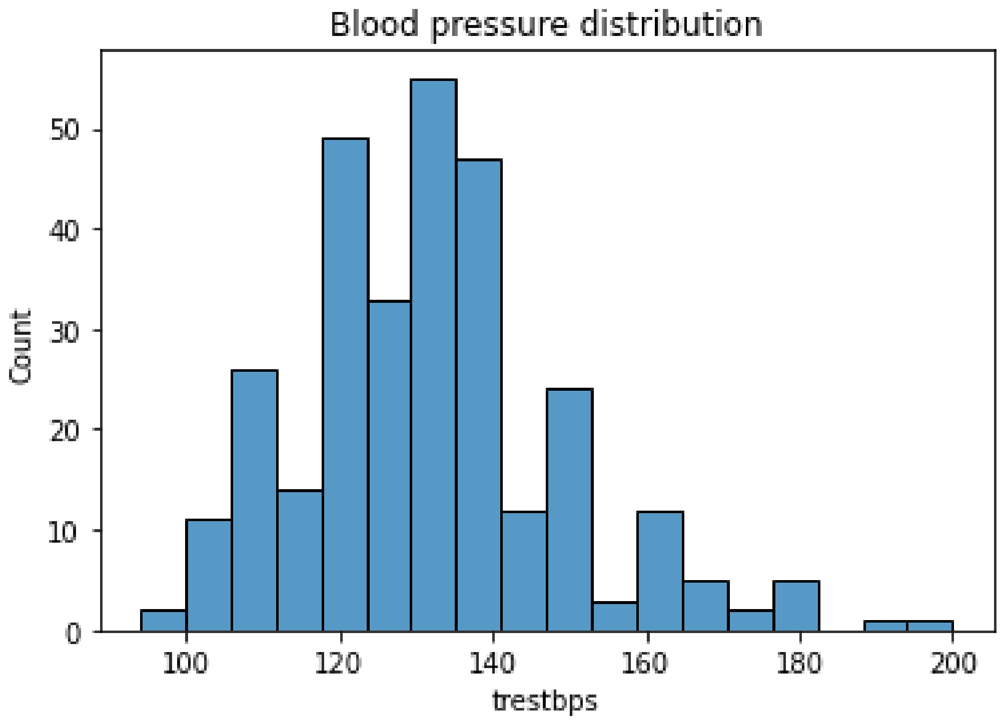
Cholesterol, mg/dl Interpretation

< 200 Desirable

200-239 Borderline

240 High

[19]:
```
sns.histplot(data.trestbps)
plt.title('Blood pressure distribution')
plt.show()
```

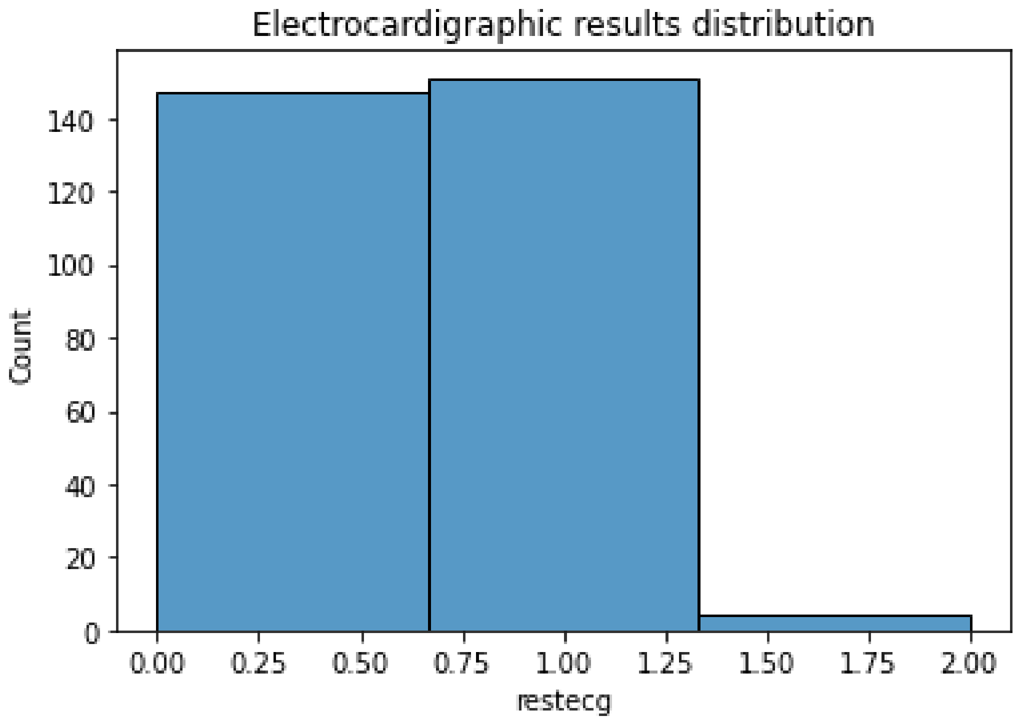


According to study, the following classification for blood pressure is applied:

```
Category            Blood pressure

Optimal                < 120

Normal                 120-129

High normal           130-139

Grade 1 hypertension    140-159

Grade 2 hypertension    160-179

Grade 3 hypertension       180
```

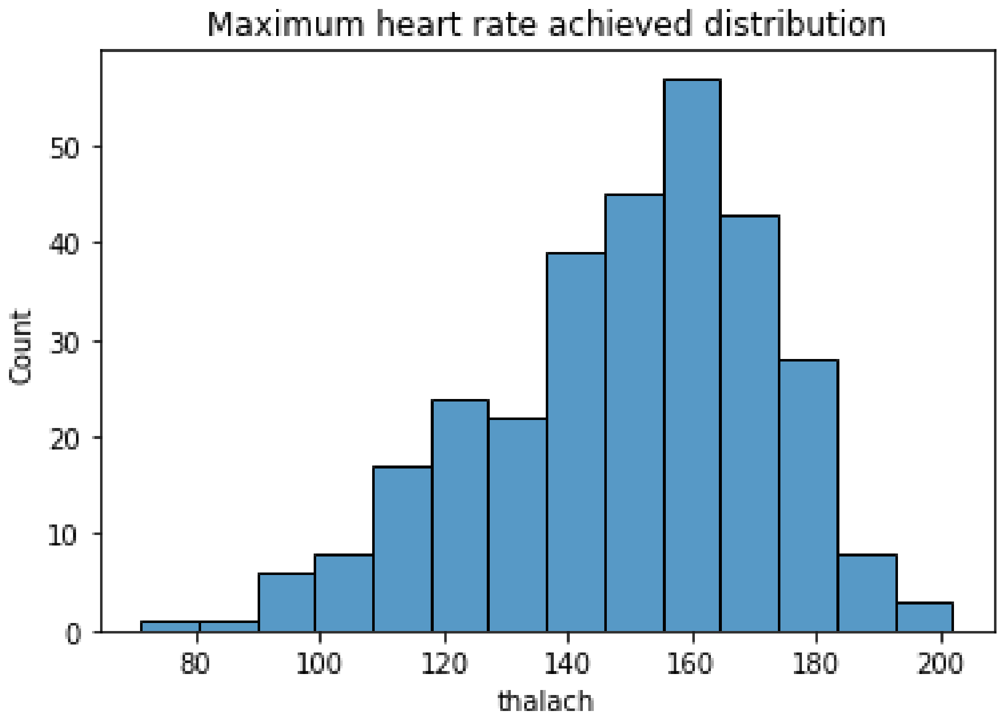
In our dataset, the resting blood pressure distribution has a peak at a value of approx. 150

[20]:
```
#Resting electrocardiographic results
sns.histplot(data.restecg, bins=3)
sns.histplot()
plt.title('Electrocardigraphic results distribution')
plt.show()
```



About 50% of the patients have hypertrophy. Only a few of the patients have ST-T wave abnormality. The rest of them have normal results.

[21]:
```
sns.histplot(data.thalach)
plt.title('Maximum heart rate achieved distribution')
plt.show()
```



Highest Heart Rate is within 170-180

[22]:
```
data.fbs.value_counts()
```

[22]:
```
0    257
1     45
Name: fbs, dtype: int64
```

From the study 0 means low sugar 1 means high sugar so maximum of them have less blood sugar in fasting than those who have sugar.

[23]:
```
from sklearn.preprocessing import MinMaxScaler

scale = MinMaxScaler()
data[['age','trestbps','chol','thalach','oldpeak']]=scale.
 ↪fit_transform(data[['age','trestbps','chol','thalach','oldpeak']])
```

[24]:
```
X=data.iloc[:,0:13]
Y=data['target']

X
```

[24]:
```
        age  sex  cp  trestbps      chol  fbs  restecg   thalach  exang  \
0  0.479167    1   0  0.292453  0.196347    0        1  0.740458      0
```

```
1     0.500000    1    0  0.433962  0.175799    1         0  0.641221       1
2     0.854167    1    0  0.481132  0.109589    0         1  0.412214       1
3     0.666667    1    0  0.509434  0.175799    0         1  0.687023       0
4     0.687500    0    0  0.415094  0.383562    1         1  0.267176       0
..         …   …   ..        …        …   …       …         …     …
723   0.812500    0    2  0.245283  0.194064    0         0  0.335878       0
733   0.312500    0    2  0.132075  0.034247    0         1  0.793893       0
739   0.479167    1    0  0.320755  0.294521    0         1  0.687023       1
843   0.625000    1    3  0.622642  0.335616    0         0  0.412214       0
878   0.520833    1    0  0.245283  0.141553    0         1  0.320611       0

      oldpeak  slope  ca  thal
0     0.161290      2   2     3
1     0.500000      0   0     3
2     0.419355      0   0     3
3     0.000000      2   1     3
4     0.306452      1   3     2
..         …      …  ..   …
723   0.241935      1   0     2
733   0.096774      1   0     2
739   0.000000      2   1     3
843   0.000000      2   0     2
878   0.225806      1   1     3

[302 rows x 13 columns]
```

[44]:
```python
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y)
```

[45]:
```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
```

[46]:
```python
model=DecisionTreeClassifier()
```

[50]:
```python
model.fit(X_train,Y_train)
ypred=model.predict(X_test)
print(classification_report(Y_test,ypred))
print("Accuracy Score",accuracy_score(Y_test,ypred)*100)
```

```
              precision    recall  f1-score   support

           0       0.83      0.73      0.77        33
           1       0.81      0.88      0.84        43

    accuracy                           0.82        76
   macro avg       0.82      0.81      0.81        76
weighted avg       0.82      0.82      0.81        76
```

```
Accuracy Score 81.57894736842105
```

[51]:
```python
from sklearn.ensemble import RandomForestClassifier
```

[54]:
```python
m1=RandomForestClassifier(n_estimators=100,criterion="gini")
```

[55]:
```python
m1.fit(X_train,Y_train)
Y1p=m1.predict(X_test)
print(classification_report(Y_test,Y1p))
print("Accuracy Score",accuracy_score(Y_test,Y1p)*100)
```

```
              precision    recall  f1-score   support

           0       0.94      0.88      0.91        33
           1       0.91      0.95      0.93        43

    accuracy                           0.92        76
   macro avg       0.92      0.92      0.92        76
weighted avg       0.92      0.92      0.92        76

Accuracy Score 92.10526315789474
```

[60]:
```python
Y1p
```

[60]:
```
array([1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 1], dtype=int64)
```

[65]:
```python
m2=RandomForestClassifier(n_estimators=200,criterion="entropy")
```

[66]:
```python
m2.fit(X_train,Y_train)
prd1=m2.predict(X_test)
print(classification_report(Y_test,prd1))
print("Accuracy Score",accuracy_score(Y_test,prd1)*100)
```

```
              precision    recall  f1-score   support

           0       0.90      0.85      0.88        33
           1       0.89      0.93      0.91        43

    accuracy                           0.89        76
   macro avg       0.90      0.89      0.89        76
weighted avg       0.90      0.89      0.89        76

Accuracy Score 89.47368421052632
```

[67]:
```
prd1
```

[67]:
```
array([1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 1], dtype=int64)
```

From the result we can state that in future there is possibility for patients to get heart attack

[ ]:
```

```