In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
df = pd.read_csv('heart.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	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

In [4]:

```
df.isnull().sum()
```

Out[4]:

```
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 [5]:

```
for feature in df.columns:
    print(f"{feature} unique values are: {df[feature].unique()}")
```

```
age unique values are: [52 53 70 61 62 58 55 46 54 71 43 34 51 50 60 67 45 6
3 42 44 56 57 59 64
 65 41 66 38 49 48 29 37 47 68 76 40 39 77 69 35 74]
sex unique values are: [1 0]
cp unique values are: [0 1 2 3]
trestbps unique values are: [125 140 145 148 138 100 114 160 120 122 112 132
118 128 124 106 104 135
 130 136 180 129 150 178 146 117 152 154 170 134 174 144 108 123 110 142
 126 192 115  94 200 165 102 105 155 172 164 156 101]
chol unique values are: [212 203 174 294 248 318 289 249 286 149 341 210 298
204 308 266 244 211
 185 223 208 252 209 307 233 319 256 327 169 131 269 196 231 213 271 263
 229 360 258 330 342 226 228 278 230 283 241 175 188 217 193 245 232 299
 288 197 315 215 164 326 207 177 257 255 187 201 220 268 267 236 303 282
 126 309 186 275 281 206 335 218 254 295 417 260 240 302 192 225 325 235
 274 234 182 167 172 321 300 199 564 157 304 222 184 354 160 247 239 246
 409 293 180 250 221 200 227 243 311 261 242 205 306 219 353 198 394 183
 237 224 265 313 340 259 270 216 264 276 322 214 273 253 176 284 305 168
 407 290 277 262 195 166 178 141]
fbs unique values are: [0 1]
restecg unique values are: [1 0 2]
thalach unique values are: [168 155 125 161 106 122 140 145 144 116 136 192
156 142 109 162 165 148
 172 173 146 179 152 117 115 112 163 147 182 105 150 151 169 166 178 132
 160 123 139 111 180 164 202 157 159 170 138 175 158 126 143 141 167  95
 190 118 103 181 108 177 134 120 171 149 154 153  88 174 114 195 133  96
 124 131 185 194 128 127 186 184 188 130  71 137  99 121 187  97  90 129
 113]
exang unique values are: [0 1]
oldpeak unique values are: [1.  3.1 2.6 0.  1.9 4.4 0.8 3.2 1.6 3.  0.7 4.2
1.5 2.2 1.1 0.3 0.4 0.6
 3.4 2.8 1.2 2.9 3.6 1.4 0.2 2.  5.6 0.9 1.8 6.2 4.  2.5 0.5 0.1 2.1 2.4
 3.8 2.3 1.3 3.5]
slope unique values are: [2 0 1]
ca unique values are: [2 0 1 3 4]
thal unique values are: [3 2 1 0]
target unique values are: [0 1]
```

Numerical Features

In [6]:

```
numerical_feature = [feature for feature in df.columns if df[feature].dtype != 'O']
print(f"Total number of numerical features are: {len(numerical_feature)}")
print(numerical_feature)
```

```
Total number of numerical features are: 14
['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exan
g', 'oldpeak', 'slope', 'ca', 'thal', 'target']
```

Discrete features

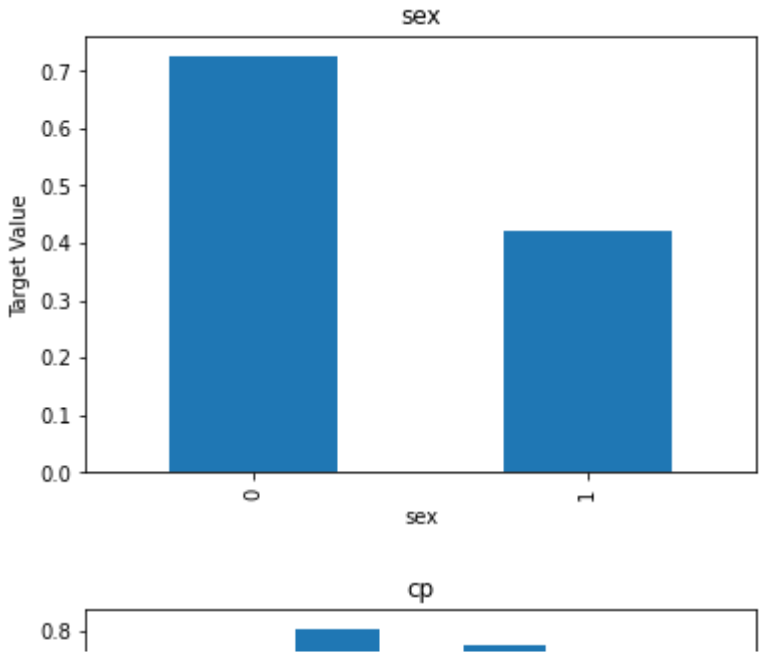
In [7]:

```
discrete_feature = [feature for feature in numerical_feature if len(df[feature].unique()) <
print(f"The total number of discrete features are {len(discrete_feature)}")
print(discrete_feature)
```

```
The total number of discrete features are 9
['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
```

In [8]:

```
for feature in discrete_feature:
    data = df.copy()
    data.groupby(feature)['target'].mean().plot.bar()
    plt.xlabel(feature)
    plt.title(feature)
    plt.ylabel('Target Value')
    plt.show()
```



In [9]:

```
df['target'].unique()
```

Out[9]:

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

In [10]:

```
filterTarget = df['target'] == 1
```

In [11]:

```
df[filterTarget]
```

Out[11]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
5	58	0	0	100	248	0	0	122	0	1.0	1	0	2	
10	71	0	0	112	149	0	1	125	0	1.6	1	0	2	
12	34	0	1	118	210	0	1	192	0	0.7	2	0	2	
15	34	0	1	118	210	0	1	192	0	0.7	2	0	2	
16	51	0	2	140	308	0	0	142	0	1.5	2	1	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1011	45	1	1	128	308	0	0	170	0	0.0	2	0	2	
1014	44	0	2	108	141	0	1	175	0	0.6	1	0	2	
1019	47	1	0	112	204	0	1	143	0	0.1	2	0	2	
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	

526 rows × 14 columns

In [12]:

```
filterNotTarget = df['target'] == 0
```

In [13]:

```
df[filterNotTarget]
```

Out[13]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1017	53	1	0	123	282	0	1	95	1	2.0	1	2	3	
1018	41	1	0	110	172	0	0	158	0	0.0	2	0	3	
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	

499 rows × 14 columns

Continous Feature

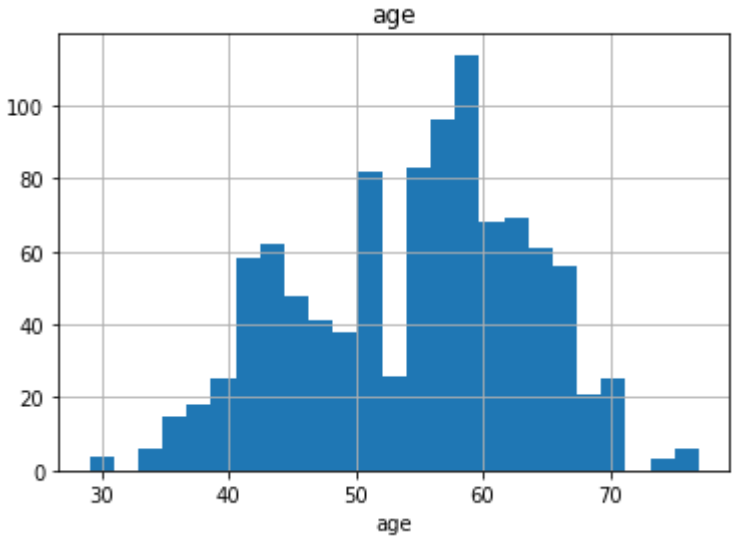
In [14]:

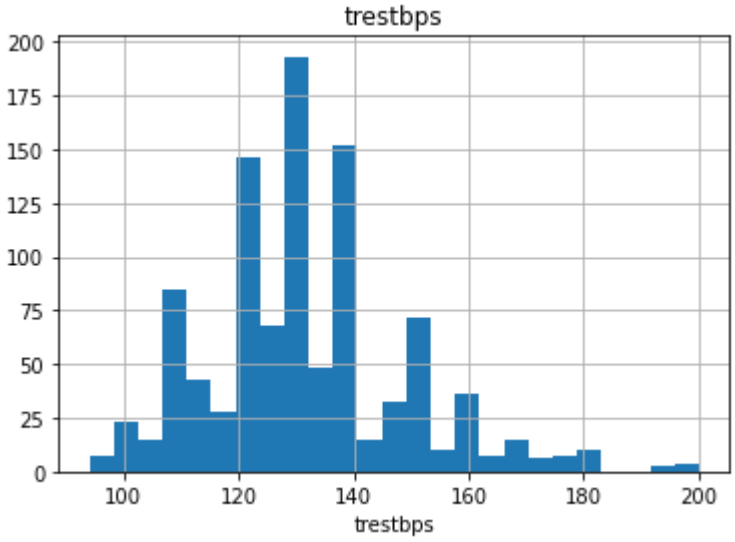
```
continous_feature = [feature for feature in df.columns if feature not in discrete_feature]

print(f"Total number of continous features are: {len(continous_feature)}")
print(continous_feature)
```

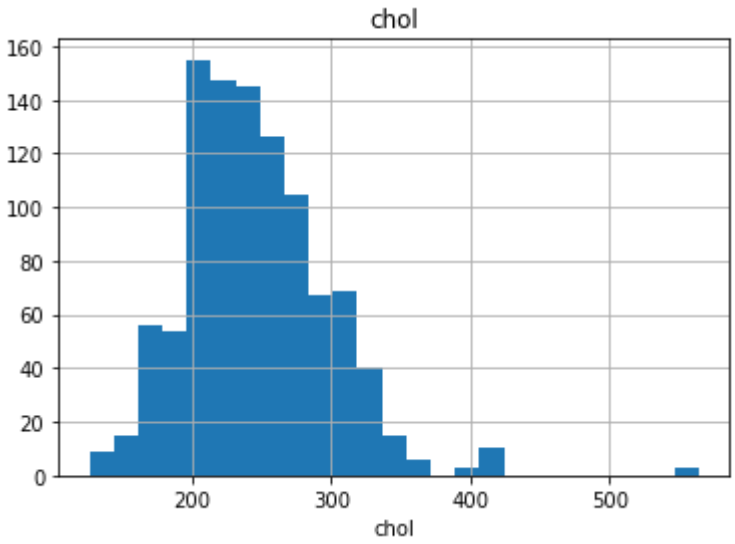
```
Total number of continous features are: 5
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

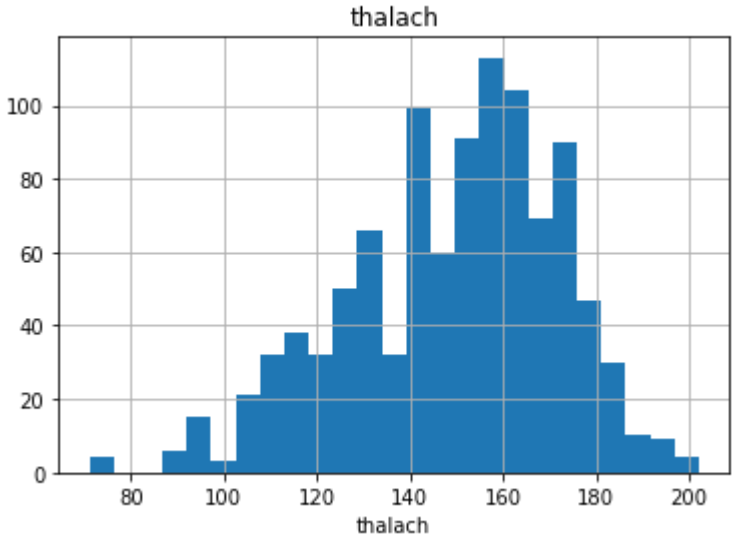
In [15]:

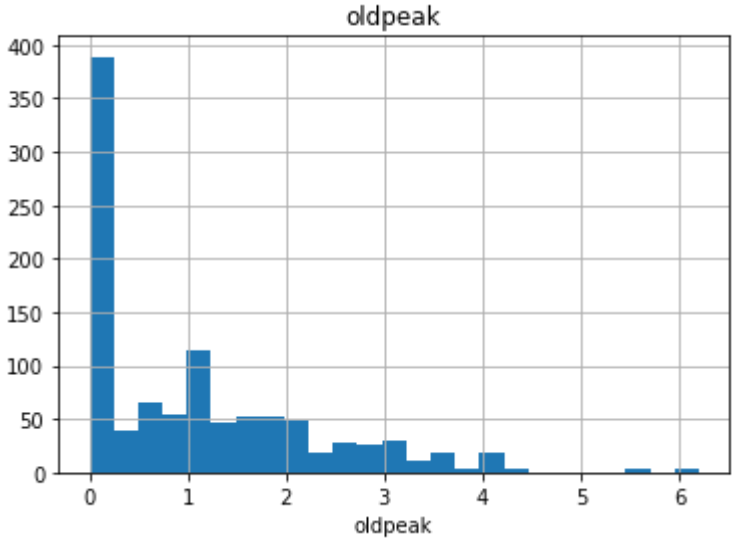
```
for feature in continous_feature:
    data = df.copy()
    data[feature].hist(bins=25)
    plt.xlabel(feature)
    plt.title(feature)
    plt.show()
```











In [16]:

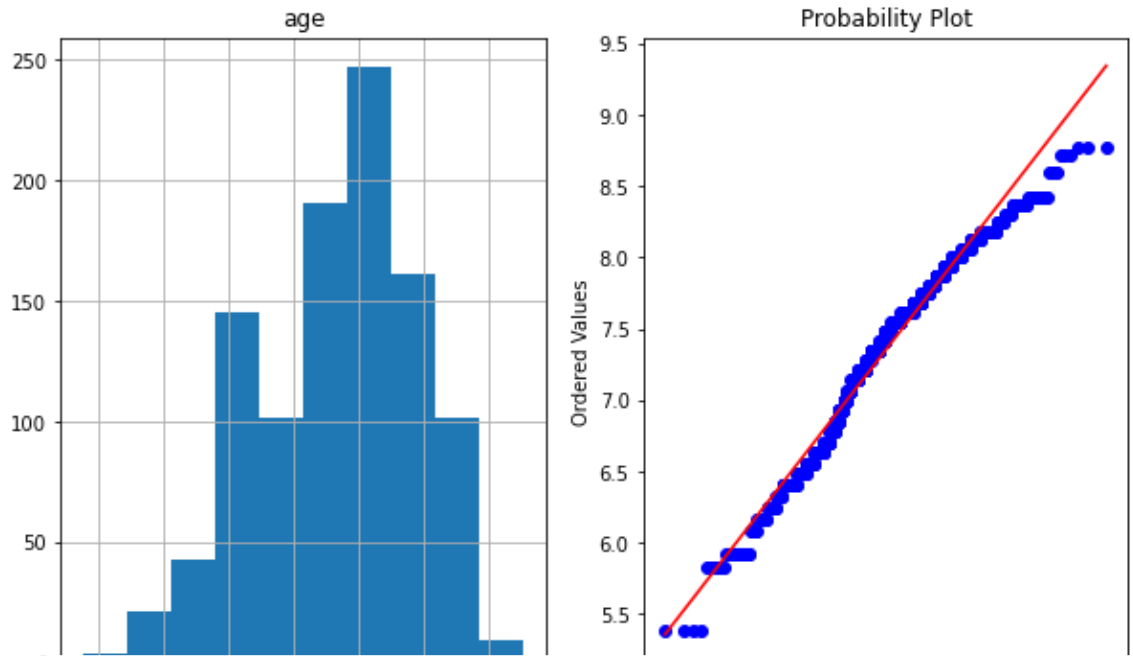
```
import pylab
import scipy.stats as stat
```

In [17]:

```
def plot_data(df, feature):
    plt.figure(figsize=(10, 6))
    plt.subplot(1, 2, 1)
    df[feature].hist()
    plt.title(feature)
    plt.subplot(1, 2, 2)
    stat.probplot(df[feature], dist='norm', plot=pylab)
    plt.show()
```

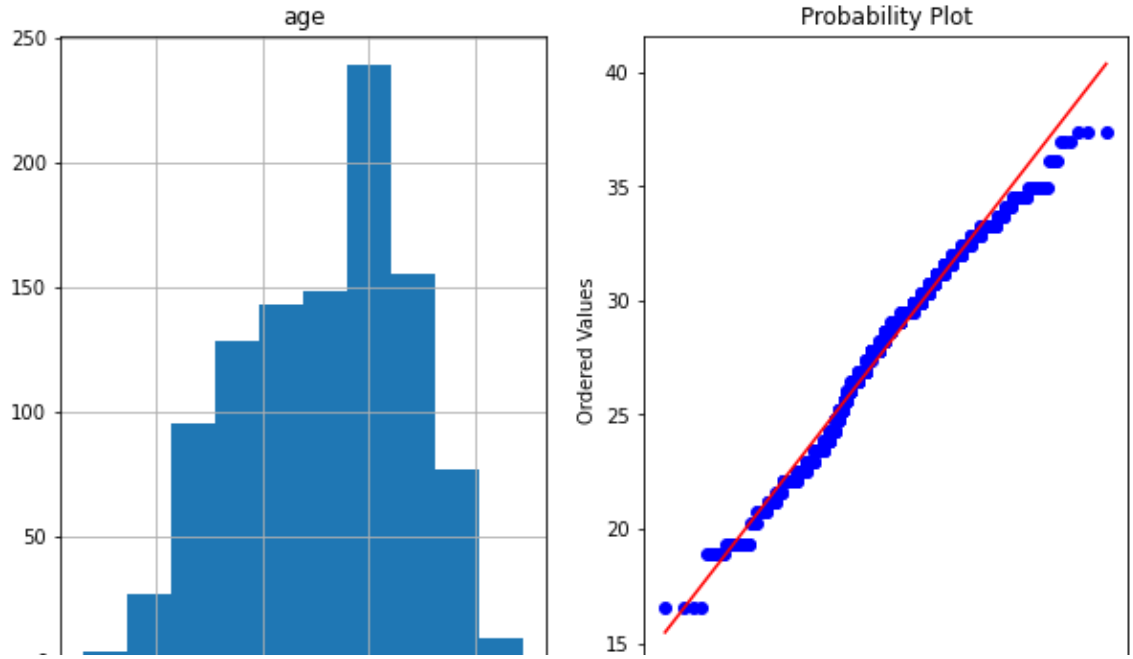
In [18]:

```
# Using Square Root Transformation
for feature in continous_feature:
    data = df.copy()
    data[feature] = data[feature] ** 0.5
    plot_data(data, feature)
```



In [19]:

```
# Using exponential Transformation
for feature in continous_feature:
    data = df.copy()
    data[feature] = (data[feature] ** (1 / 1.2))
    plot_data(data, feature)
```



In [20]:

```
# Using Exponential Transformation for the training dataset
for feature in continous_feature:
    df[feature] = (df[feature] ** (1 / 1.2))
```

In [21]:

```
df.head()
```

Out[21]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
0	26.915521	1	0	55.901699	86.818688	0	1	71.519489	0	1.000000	2
1	27.346173	1	0	61.438419	83.736222	1	0	66.876943	1	2.567250	0
2	34.481147	1	0	63.261575	73.641796	0	1	55.901699	1	2.217232	0
3	30.745026	1	0	64.350426	83.736222	0	1	69.027408	0	0.000000	2
4	31.164470	0	0	60.706133	114.013456	1	1	48.725341	0	1.707240	1

In [22]:

```
X = df.drop(['target'], axis=1)
```

In [23]:

```
X.head()
```

Out[23]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
0	26.915521	1	0	55.901699	86.818688	0	1	71.519489	0	1.000000	2
1	27.346173	1	0	61.438419	83.736222	1	0	66.876943	1	2.567250	0
2	34.481147	1	0	63.261575	73.641796	0	1	55.901699	1	2.217232	0
3	30.745026	1	0	64.350426	83.736222	0	1	69.027408	0	0.000000	2
4	31.164470	0	0	60.706133	114.013456	1	1	48.725341	0	1.707240	1

In [24]:

```
Y = df['target']
```

In [25]:

```
Y
```

Out[25]:

```
0       0
1       0
2       0
3       0
4       0
       ..
1020    1
1021    0
1022    0
1023    1
1024    0
Name: target, Length: 1025, dtype: int64
```

In [26]:

```
X = np.array(X)
```

In [27]:

```
Y = np.array(Y)
```

In [28]:

```
X.shape, Y.shape
```

Out[28]:

```
((1025, 13), (1025,))
```

In [29]:

```
df.shape
```

Out[29]:

```
(1025, 14)
```

In [30]:

```
from sklearn.model_selection import train_test_split
```

In [31]:

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, shuffle=True)
```

In [32]:

```
X_train.shape, y_train.shape
```

Out[32]:

```
((768, 13), (768,))
```

In [33]:

```
X_test.shape, y_test.shape
```

Out[33]:

```
((257, 13), (257,))
```

In [34]:

```
n_estimators = [int(x) for x in np.linspace(start=200, stop=2000, num=10)]
criterion = ["gini", "entropy", "log_loss"]
min_samples_leaf = [1 ,2 , 4, 6, 8, 10]
max_features = ["sqrt", "log2", None]
min_samples_split = [2, 5, 6, 8, 10, 12]
```

In [35]:

```
random_grid = {
    'n_estimators' : n_estimators,
    'criterion' : criterion,
    'min_samples_split' : min_samples_split,
    'min_samples_leaf' : min_samples_leaf,
    'max_features' : max_features
}
```

In [36]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [37]:

```
rf = RandomForestClassifier()
```

In [38]:

```
from sklearn.model_selection import KFold
```

In [39]:

```
cv = KFold(n_splits=5, random_state=True, shuffle=True)
```

In [40]:

```
from sklearn.model_selection import RandomizedSearchCV
```

In [41]:

```
rf_randomcv = RandomizedSearchCV(estimator=rf,param_distributions=random_grid, n_iter=100,
```

In [42]:

```
rf_randomcv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
```

Out[42]:

```
RandomizedSearchCV(cv=KFold(n_splits=5, random_state=True, shuffle=True),
                   estimator=RandomForestClassifier(), n_iter=100, n_jobs=-
1,
                   param_distributions={'criterion': ['gini', 'entropy',
                                                      'log_loss'],
                                        'max_features': ['sqrt', 'log2', Non
e],
                                        'min_samples_leaf': [1, 2, 4, 6, 8,
10],
                                        'min_samples_split': [2, 5, 6, 8, 1
0,
                                                              12],
                                        'n_estimators': [200, 400, 600, 800,
                                                         1000, 1200, 1400, 1
600,
                                                         1800, 2000]},
                   random_state=100, verbose=2)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [43]:

```
best_params = rf_randomcv.best_params_
```

In [44]:

```
best_params
```

Out[44]:

```
{'n_estimators': 2000,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': None,
 'criterion': 'log_loss'}
```

In [45]:

```
from sklearn.model_selection import GridSearchCV
```

In [46]:

```
param_grid = {
    'criterion' : [best_params['criterion']],
    'min_samples_leaf' : [best_params['min_samples_leaf']-2, best_params['min_samples_leaf'
                          best_params['min_samples_leaf'], best_params['min_samples_leaf']+1
                          best_params['min_samples_leaf']+2],
    'min_samples_split' : [best_params['min_samples_split']-2, best_params['min_samples_spl
                           best_params['min_samples_split'], best_params['min_samples_split'
                           best_params['min_samples_split']+2],
    'max_features' : [best_params['max_features']],
    'n_estimators' : [best_params['n_estimators']-200, best_params['n_estimators']-100,
                      best_params['n_estimators'], best_params['n_estimators']+100,
                      best_params['n_estimators']+200]
}
```

In [47]:

```
rf = RandomForestClassifier()
```

In [48]:

```
grid_searchcv = GridSearchCV(estimator=rf, param_grid=param_grid, cv=cv,n_jobs=-1, verbose=
```

In [49]:

```
grid_searchcv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 125 candidates, totalling 625 fits
```

In [50]:

```
best_grid = grid_searchcv.best_estimator_
```

In [51]:

```
best_grid
```

Out[51]:

```
RandomForestClassifier(criterion='log_loss', max_features=None,
                       n_estimators=1800)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [52]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

In [53]:

```
y_pred = best_grid.predict(X_test)
```

In [54]:

```
print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[121   3]
 [  0 133]]
0.9883268482490273
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       124
           1       0.98      1.00      0.99       133

    accuracy                           0.99       257
   macro avg       0.99      0.99      0.99       257
weighted avg       0.99      0.99      0.99       257
```

In [55]:

```
pip install pandoc
```

```
Defaulting to user installation because normal site-packages is not writeabl
e
Requirement already satisfied: pandoc in c:\users\19138\appdata\roaming\pyth
on\python39\site-packages (2.2)
Requirement already satisfied: plumbum in c:\users\19138\appdata\roaming\pyt
hon\python39\site-packages (from pandoc) (1.8.0)
Requirement already satisfied: ply in c:\users\19138\appdata\roaming\python
\python39\site-packages (from pandoc) (3.11)
Requirement already satisfied: pywin32 in c:\programdata\anaconda3\lib\site-
packages (from plumbum->pandoc) (302)
Note: you may need to restart the kernel to use updated packages.
```

In [58]:

```
import pickle
pickle.dump(best_grid, open('RandomForestClassifier', 'wb'))
```

In [59]:

```
loaded_model = pickle.load(open('RandomForestClassifier', 'rb'))
```

In [60]:

```
loaded_model
```

Out[60]:

```
RandomForestClassifier(criterion='log_loss', max_features=None,
                       n_estimators=1800)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [62]:

```
y_load_pred = loaded_model.predict(X_test)
```

In [63]:

```
print(confusion_matrix(y_test, y_load_pred))
print(accuracy_score(y_test, y_load_pred))
print(classification_report(y_test, y_load_pred))
```

```
[[121   3]
 [  0 133]]
0.9883268482490273
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       124
           1       0.98      1.00      0.99       133

    accuracy                           0.99       257
   macro avg       0.99      0.99      0.99       257
weighted avg       0.99      0.99      0.99       257
```

In [64]:

```
y_train_pred = loaded_model.predict(X_train)
```

In [66]:

```
print(accuracy_score(y_train, y_train_pred))
print(confusion_matrix(y_train, y_train_pred))
print(classification_report(y_train, y_train_pred))
```

```
1.0
[[375   0]
 [  0 393]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       375
           1       1.00      1.00      1.00       393

    accuracy                           1.00       768
   macro avg       1.00      1.00      1.00       768
weighted avg       1.00      1.00      1.00       768
```

In [67]:

```
test_data = pd.read_csv('heart1.csv')
```

In [68]:

```
test_data.head()
```

Out[68]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [69]:

```
for feature in df.columns:
    print(f"{feature} : {df[feature].unique()}")
```

```
age : [26.91552072 27.34617307 34.48114668 30.74502585 31.1644697  29.479718
22
 28.20345029 24.30141301 27.7754732  34.89115054 22.97331741 18.89002057
 26.48348578 26.05003655 30.32443435 33.245204   23.86036565 31.58278747
 22.52722735 23.41768162 28.63013236 29.05554637 29.90267287 32.
 32.4161274  22.07936339 32.83118913 20.72460828 25.61513997 25.17876152
 16.54493245 20.26911422 24.74086512 33.65819023 36.92705389 21.62967494
 21.17810867 37.33151408 34.07016543 19.35188927 36.11545576]
sex : [1 0]
cp : [0 1 2 3]
trestbps : [55.90169944 61.43841876 63.26157524 64.35042648 60.70613341 46.4
1588834
 51.77109802 68.66993751 54.03199939 54.78140824 51.01309644 58.49852862
 53.28050585 57.01751796 55.52877221 48.72534106 47.95800582 59.60437528
 57.7589727  59.97207705 75.75193907 57.38848482 65.07428163 75.04987996
 63.62493912 52.90396448 65.79652996 66.51719609 72.22830686 59.23621926
 73.64179632 62.89779346 49.49026693 55.1553434  50.25283541 62.16896258
 56.27412975 79.93758505 52.14926602 44.08321004 82.70371084 70.45361581
 47.18820707 48.34197795 66.87694334 72.93573644 70.09760934 67.23630397
 46.80236632]
chol : [ 86.81868779  83.73622246  73.64179632 114.01345557  98.94102397
 121.71831548 112.39531331  99.27337555 111.42219103  64.71255648
 129.01156666  86.13561237 115.30466722  84.07982637 118.52016946
 104.89014776  97.60937275  86.47728497  77.50144032  90.55672738
  85.45145182 100.26910012  85.79366849 118.19941113  93.92833
 122.03720033 101.59366733 124.58233331  71.87407258  58.12898589
 105.87503494  81.32300027  93.25597132  87.15982231 106.53060883
 103.9034075   92.58264172 134.97461084 102.25465512 125.53407204
 129.32676724  91.57080511  92.24560973 108.81881608  92.9194285
 110.44736596  96.6082471   73.99431838  78.54735063  88.52170663
  80.2843861   97.94262442  93.59227142 115.62701752 112.07112714
  81.66861461 120.76065543  87.84129234  70.09760934 124.2647642
  85.10896083  74.69835799 101.92426839 101.26285096  78.1990251
  83.04816633  89.54037331 105.54694361 105.21864818  94.93506873
 116.91462933 110.12204225  56.27412975 118.84075426  77.85038898
 107.83934587 109.79652621  84.76619397 127.1171086   88.86152143
 100.9318183  114.33653149 152.5621191  102.91478947  96.27407799
 116.59299339  79.93758505  91.23302993 123.94703269  94.59972746
 107.51246081  94.26414828  76.45269926  71.16455303  72.93573644
 122.67447094 115.94918818  82.35896818 196.21517256  67.59528088
 117.23608839  90.21819738  77.15217714 133.09734276  68.66993751
  98.60844896  95.93967673  98.27564947 150.11914769 113.69019646
  75.75193907  99.60550474  89.87941314  82.70371084  91.90833128
  97.27589337 119.48140591 103.244539    96.94218519  84.42314968
 117.87847862  89.20107653 132.78395126  82.01393667 145.51692814
  76.80259746  95.27017325  90.89500446 104.56144144 120.12137126
 128.696212   102.5848285  106.20292306  88.18163074 104.23252832
 108.16603288 122.99285791  87.50069001 107.18537685 100.60056835
  74.34650486 110.77249814 117.55737126  71.51948877 149.50716546
 112.71931259 108.49252267 103.57407803  80.97709191  70.80926287
  75.04987996  61.80390655]
fbs : [0 1]
restecg : [1 0 2]
thalach : [71.51948877 66.87694334 55.90169944 69.02740761 48.72534106 54.78
140824
```

```
 61.43841876 63.26157524 62.89779346 52.52688634 59.97207705 79.93758505
 67.23630397 62.16896258 49.87184268 69.38450784 70.45361581 64.35042648
 72.93573644 73.28893643 63.62493912 75.40107294 65.79652996 52.90396448
 52.14926602 51.01309644 69.74124088 63.98788844 76.45269926 48.34197795
 65.07428163 65.43560509 71.87407258 70.80926287 75.04987996 58.49852862
 68.66993751 55.1553434  61.0724956  50.63325132 75.75193907 70.09760934
 83.39233632 67.59528088 68.31209486 72.22830686 60.70613341 73.99431838
 67.9538769  56.27412975 62.53359039 61.80390655 71.16455303 44.4736736
 79.24307761 53.28050585 47.57341786 76.10248048 49.49026693 74.69835799
 59.23621926 54.03199939 72.58219403 64.71255648 66.51719609 66.15705928
 41.72556507 73.64179632 51.77109802 80.97709191 58.86760505 44.86345273
 55.52877221 58.12898589 77.50144032 80.63088778 57.01751796 56.64606775
 77.85038898 77.15217714 78.54735063 57.7589727  34.89115054 60.33932849
 46.02876567 54.40696188 78.1990251  45.25255573 42.51433835 57.38848482
 51.39237674]
exang : [0 1]
oldpeak : [1.         2.5672495  2.21723193 0.         1.70724035 3.43724559
 0.83031265 2.63607836 1.47944896 2.49804953 0.74287379 3.30654478
 1.4019829  1.92908886 1.08266452 0.36666353 0.46599722 0.65332013
 2.77267584 2.35847726 1.16408414 2.42846397 2.90793984 1.32365061
 0.2615321  1.78179744 4.20232873 0.91594365 1.63202605 4.57431858
 3.1748021  2.14593555 0.56123102 0.14677993 1.85573551 2.07416213
 3.04195648 2.0018884  1.24437934 2.84046889]
slope : [2 0 1]
ca : [2 0 1 3 4]
thal : [3 2 1 0]
target : [0 1]
```

In [78]:

```
numerical_feature = [feature for feature in test_data.columns if test_data[feature].dtype !
print(numerical_feature)
```

```
['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh', 'exng',
'oldpeak', 'slp', 'caa', 'thall', 'output']
```

In [79]:

```
continous_feature = [feature for feature in numerical_feature if len(test_data[feature].uni
print(continous_feature)
```

```
['age', 'trtbps', 'chol', 'thalachh', 'oldpeak']
```

In [80]:

```
continous_feature
```

Out[80]:

```
['age', 'trtbps', 'chol', 'thalachh', 'oldpeak']
```

In [81]:

```
for feature in continous_feature:
    test_data[feature] = test_data[feature] ** (1 / 1.2)
```

In [82]:

```
test_data.head()
```

Out[82]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	ca
0	17.764053	1	3	63.261575	93.928330	1	0	65.074282	0	2.001888	0	
1	12.275211	1	2	57.758973	99.605505	0	1	78.199025	0	2.840469	0	
2	13.182227	0	1	57.758973	84.079826	0	0	72.935736	0	1.323651	2	
3	16.368899	1	1	54.031999	94.935069	0	1	75.049880	0	0.830313	2	
4	16.571337	0	0	54.031999	133.097343	0	1	69.741241	1	0.653320	2	

In [83]:

```
Xnew_test = np.array(test_data.drop(['output'], axis=1))
```

In [84]:

```
Xnew_test.shape
```

Out[84]:

```
(303, 13)
```

In [85]:

```
Ynew_test = np.array(test_data['output'])
```

In [87]:

```
Ynew_test.shape
```

Out[87]:

```
(303,)
```

In [88]:

```
y_new_pred = loaded_model.predict(Xnew_test)
```

In [90]:

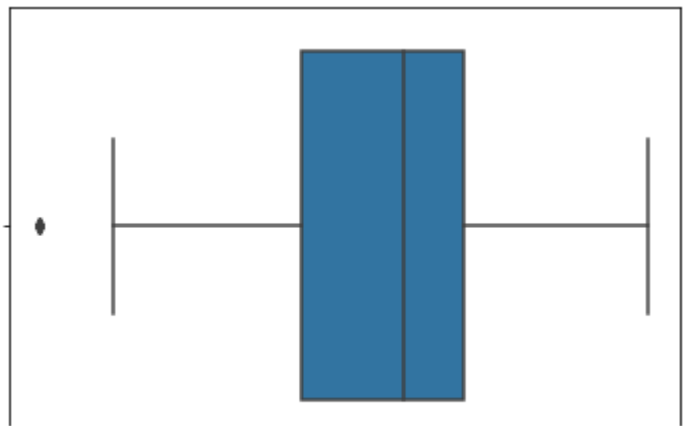
```
print(accuracy_score(Ynew_test, y_new_pred))
print(confusion_matrix(Ynew_test, y_new_pred))
print(classification_report(Ynew_test, y_new_pred))
```

```
0.9273927392739274
[[122  16]
 [  6 159]]
              precision    recall  f1-score   support

           0       0.95      0.88      0.92       138
           1       0.91      0.96      0.94       165

    accuracy                           0.93       303
   macro avg       0.93      0.92      0.93       303
weighted avg       0.93      0.93      0.93       303
```

In [92]:

```
import seaborn as sns
for feature in df.columns:
    data = df.copy()
    sns.boxplot(data[feature])
    plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```



In [93]:

```
best_grid
```

Out[93]:

```
RandomForestClassifier(criterion='log_loss', max_features=None,
                       n_estimators=1800)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [97]:

```
pickle.dump(best_grid, open('model.pkl', 'wb'))
```

In [100]:

```
new_loaded_model = pickle.load(open('model.pkl', "rb"))
```

In [101]:

```
new_loaded_model
```

Out[101]:

```
RandomForestClassifier(criterion='log_loss', max_features=None,
                       n_estimators=1800)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [102]:

```
new_loaded_model.predict(X_train)
```

Out[102]:

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

In [105]:

```
test_data = pd.read_csv('heart1.csv')
```

In [106]:

```
test_data.head()
```

Out[106]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [107]:

```
numerical_feature = [feature for feature in test_data.columns if test_data[feature].dtype !
```

In [108]:

```
continous_feature = [feature for feature in numerical_feature if len(test_data[feature].uni
```

In [109]:

```
continous_feature
```

Out[109]:

```
['age', 'trtbps', 'chol', 'thalachh', 'oldpeak']
```

In [110]:

```
for feature in continous_feature:
    test_data[feature] = test_data[feature] ** (1 / 1.2)
```

In [112]:

```
test_data.head(15)
```

Out[112]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	c
0	31.582787	1	3	63.261575	93.928330	1	0	65.074282	0	2.001888	0	
1	20.269114	1	2	57.758973	99.605505	0	1	78.199025	0	2.840469	0	
2	22.079363	0	1	57.758973	84.079826	0	0	72.935736	0	1.323651	2	
3	28.630132	1	1	54.031999	94.935069	0	1	75.049880	0	0.830313	2	
4	29.055546	0	0	54.031999	133.097343	0	1	69.741241	1	0.653320	2	
5	29.055546	1	0	61.438419	79.937585	0	1	64.350426	0	0.465997	1	
6	28.630132	0	1	61.438419	114.013456	0	0	66.157059	0	1.244379	1	
7	23.417682	1	1	54.031999	103.903407	0	1	73.288936	0	0.000000	2	
8	26.915521	1	2	72.935736	82.358968	1	1	69.384508	0	0.561231	2	
9	29.055546	1	2	65.074282	71.519489	0	1	73.641796	0	1.479449	2	
10	27.775473	1	0	61.438419	95.939677	0	1	68.669938	0	1.164084	2	
11	25.178762	0	2	57.758973	107.839346	0	1	61.072496	0	0.261532	2	
12	25.615140	1	1	57.758973	104.890148	0	1	72.582194	0	0.653320	2	
13	32.000000	1	3	50.252835	86.477285	0	0	62.897793	1	1.632026	1	
14	29.479718	0	3	65.074282	110.447366	1	0	69.384508	0	1.000000	2	

In [113]:

```
test_data.tail(10)
```

Out[113]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp
293	33.245204	1	2	65.796530	86.818688	0	0	65.074282	0	0.830313	1
294	23.417682	1	0	54.031999	71.874073	0	1	62.897793	1	2.358477	0
295	31.582787	1	0	61.438419	78.199025	0	0	62.897793	1	3.174802	2
296	31.582787	0	0	55.528772	81.668615	0	1	59.972077	1	0.000000	1
297	29.902673	1	0	70.097609	74.346505	1	0	42.514338	0	1.000000	1
298	29.055546	0	0	61.438419	96.608247	0	1	55.155343	1	0.261532	1
299	23.860366	1	3	50.252835	104.232528	0	1	58.498529	0	1.164084	1
300	33.658190	1	0	62.897793	80.284386	1	1	61.803907	0	2.772676	1
301	29.055546	1	0	57.758973	58.128986	0	1	52.149266	1	1.164084	1
302	29.055546	0	1	57.758973	94.935069	0	0	73.641796	0	0.000000	1

In [115]:

```
df.head(10)
```

Out[115]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
0	15.547802	1	0	28.588246	41.258270	0	1	35.103763	0	1.000000	2
1	15.754833	1	0	30.929058	40.033891	1	0	33.194313	1	2.193933	0
2	19.112526	1	0	31.692026	35.969713	0	1	28.588246	1	1.941672	0
3	17.370503	1	0	32.145945	40.033891	0	1	34.081443	0	0.000000	2
4	17.567762	0	0	30.621548	51.776190	1	1	25.495434	0	1.561633	1
5	16.772692	0	0	24.484367	46.005914	0	0	28.110010	0	1.000000	1
6	16.772692	1	0	26.816750	54.675984	0	2	30.929058	0	2.797964	0
7	16.165353	1	0	33.934299	51.163098	0	0	31.692026	1	0.856449	1
8	14.278840	1	0	27.789189	46.134660	0	0	31.540084	0	0.856449	2
9	15.960674	1	0	28.110010	50.793688	0	0	27.142596	1	2.242842	1

In [116]:

```
data = pd.read_csv('heart.csv')
```

In [117]:

```
data.head(10)
```

Out[117]:

	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
5	58	0	0	100	248	0	0	122	0	1.0	1	0	2	1
6	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
7	55	1	0	160	289	0	0	145	1	0.8	1	1	3	0
8	46	1	0	120	249	0	0	144	0	0.8	2	0	3	0
9	54	1	0	122	286	0	0	116	1	3.2	1	2	2	0

In [118]:

```
filter1 = df['target'] == 1
```

In [122]:

```
data[filter1].head(10)
```

Out[122]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
5	58	0	0	100	248	0	0	122	0	1.0	1	0	2	
10	71	0	0	112	149	0	1	125	0	1.6	1	0	2	
12	34	0	1	118	210	0	1	192	0	0.7	2	0	2	
15	34	0	1	118	210	0	1	192	0	0.7	2	0	2	
16	51	0	2	140	308	0	0	142	0	1.5	2	1	2	
18	50	0	1	120	244	0	1	162	0	1.1	2	0	2	
19	58	1	2	140	211	1	0	165	0	0.0	2	0	2	
21	67	0	0	106	223	0	1	142	0	0.3	2	2	2	
22	45	1	0	104	208	0	0	148	1	3.0	1	0	2	
23	63	0	2	135	252	0	0	172	0	0.0	2	0	2	

In [ ]:

```
```