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

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

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 303,\n  \"fields\": [\
n    {\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 9,\n        \"min\": 29,\n
\"max\": 77,\n        \"num_unique_values\": 41,\n        \"samples\":
[\n          46,\n          66,\n          48\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"sex\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"cp\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 3,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          2,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"trestbps\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 17,\n        \"min\": 94,\n
\"max\": 200,\n        \"num_unique_values\": 49,\n
\"samples\": [\n          104,\n          123\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 51,\n        \"min\": 126,\n
\"max\": 564,\n        \"num_unique_values\": 152,\n
\"samples\": [\n          277,\n          169\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fbs\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"restecg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 2,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"thalach\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 22,\n        \"min\": 71,\n
\"max\": 202,\n        \"num_unique_values\": 91,\n
\"samples\": [\n          159,\n          152\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"exang\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\":
0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"oldpeak\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
1.1610750220686343,\n        \"min\": 0.0,\n        \"max\": 6.2,\n
\"num_unique_values\": 40,\n        \"samples\": [\n          1.9,\n
3.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"slope\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 2,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          0,\n
2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ca\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 0,\n        \"max\": 4,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          2,\n
4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"thal\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 3,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"target\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

age: the age of the patient in years.

sex: the sex of the patient (1 = male, 0 = female).

cp: the type of chest pain the patient experienced (1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic).

trestbps: the resting blood pressure of the patient in mm Hg.

chol: the serum cholesterol level of the patient in mg/dl.

fbs: the fasting blood sugar level of the patient, measured in mg/dl (1 = high, 0 = low).

restecg: the resting electrocardiographic results of the patient (0 = normal, 1 = ST-T wave abnormality, 2 = left ventricular hypertrophy).

(Resting electrocardiographic (ECG or EKG) is a non-invasive diagnostic test that records the electrical activity of the heart while the patient is at rest. The test is performed using an electrocardiogram machine, which records the electrical signals produced by the heart through electrodes placed on the chest, arms, and legs.)

thalach: the maximum heart rate achieved by the patient during exercise. exang: whether the patient experienced exercise-induced angina (1 = yes, 0 = no).

oldpeak: the ST depression induced by exercise relative to rest. slope: the slope of the ST segment during peak exercise (1 = upsloping, 2 = flat, 3 = downsloping).

(ST depression induced by exercise relative to rest Oldpeak, also known as ST depression, is a common parameter measured during an exercise stress test to evaluate the presence and severity of coronary artery disease. It represents the amount of ST segment depression that occurs on an electrocardiogram (ECG) during exercise compared to rest.)

ca: the number of major vessels colored by fluoroscopy (0-3).

(he number of major vessels (0-3) colored by fluoroscopy is a parameter that is used to assess the severity of coronary artery disease (CAD) in patients who undergo coronary angiography)

thal: the type of thallium scan performed on the patient (1 = fixed defect, 2 = reversible defect, 3 = normal).

target: the presence of heart disease in the patient (0 = no disease, 1 = disease present).

```
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
```

```
#EDA
# 1.Null values
# 2.Duplicates
# 3.outliers
# 4.Label encoder

#1
df.isnull().sum()
```

```
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
```

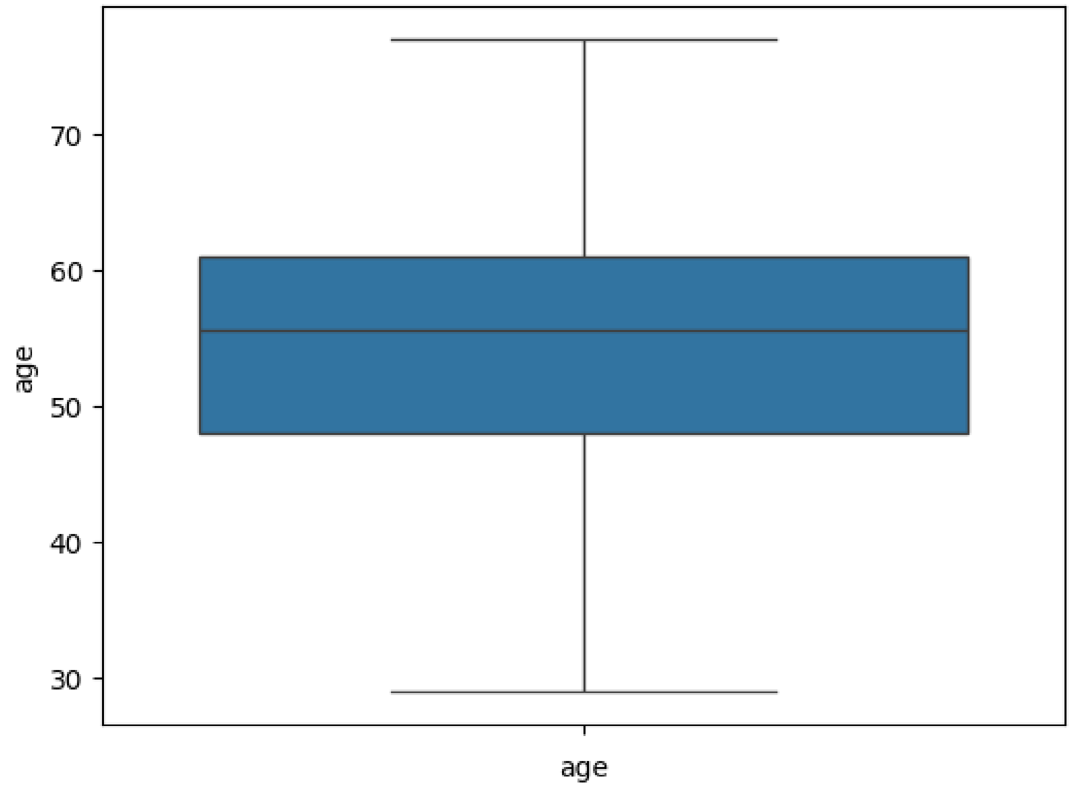
```
df.dropna(inplace=True)
```

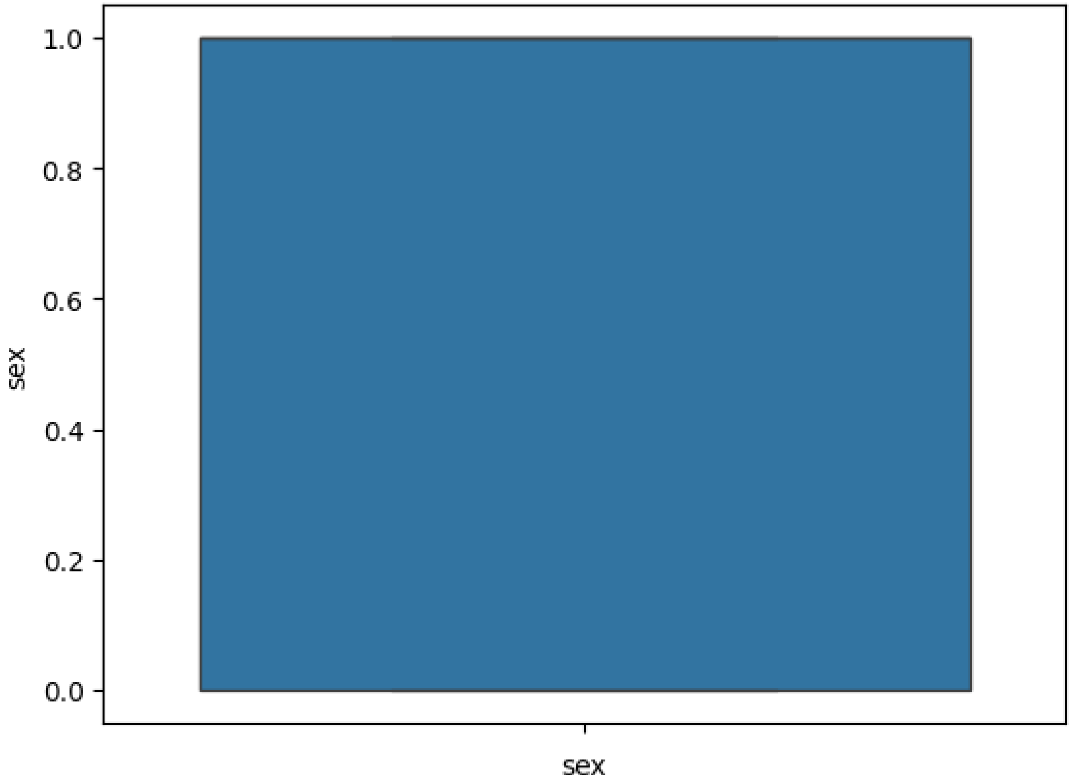
```
df.duplicated().sum()
```

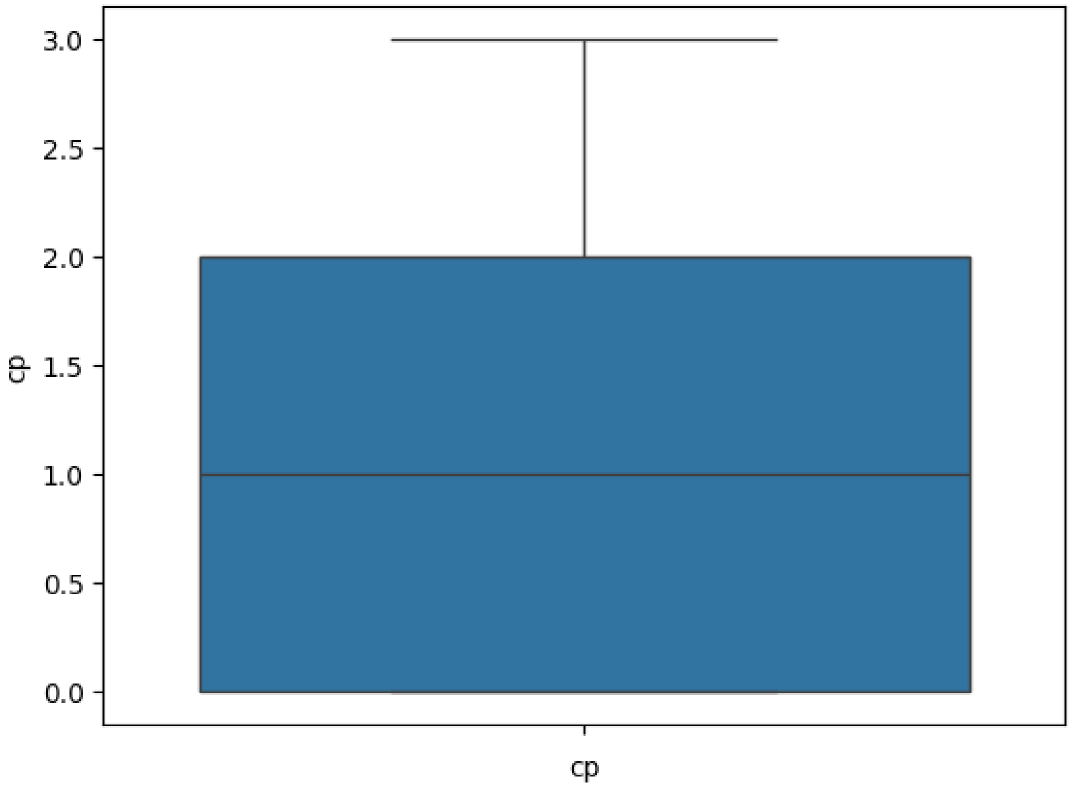
```
np.int64(1)
```

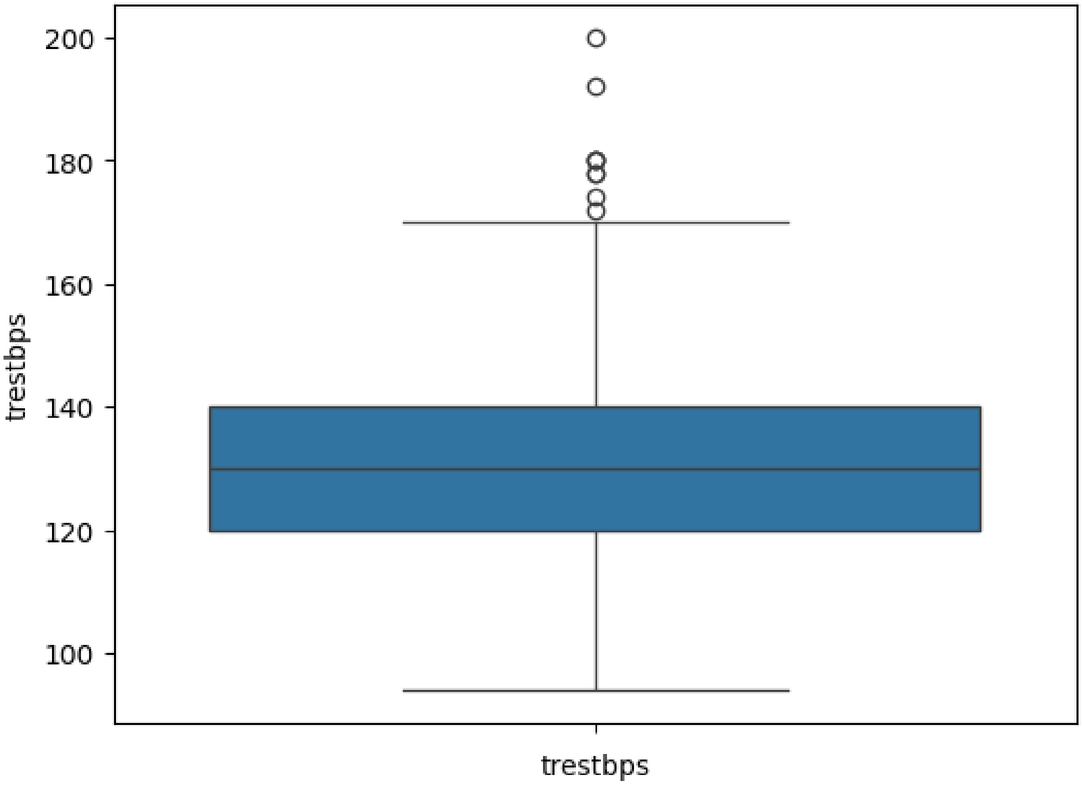
```
df.drop_duplicates(inplace=True)
```

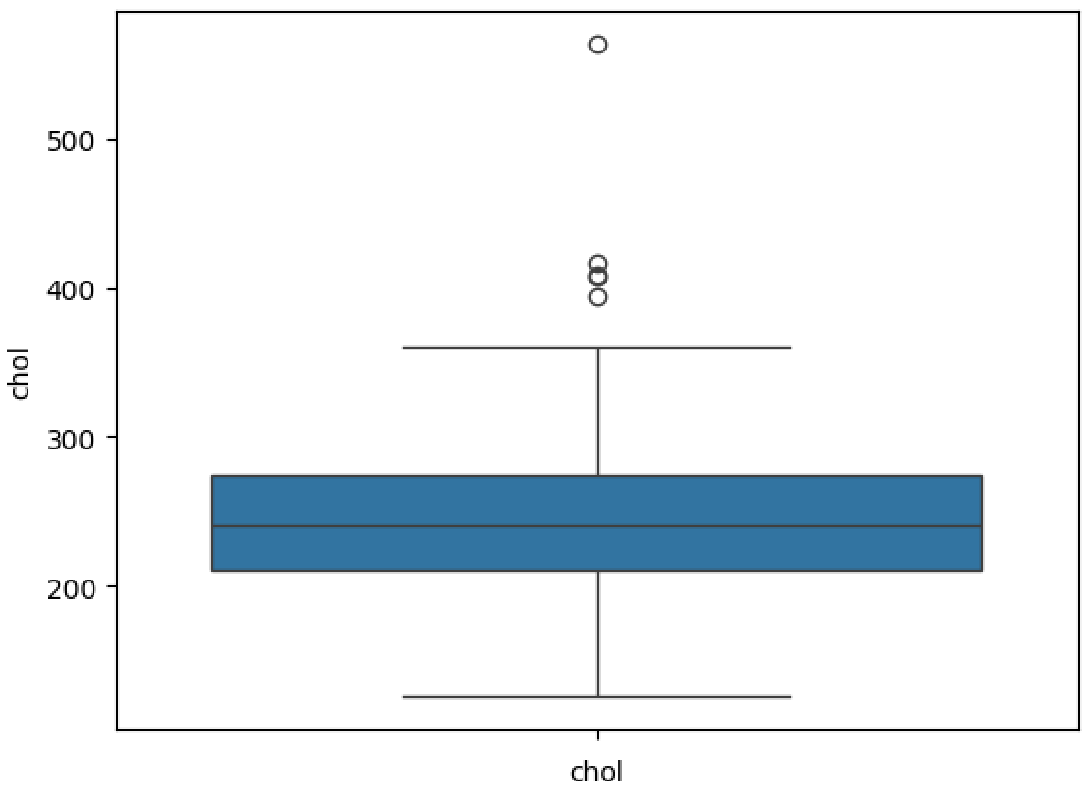
```
#outliers

for col in df.columns:
  sns.boxplot(df[col])
  plt.xlabel(col)
  plt.show()
```

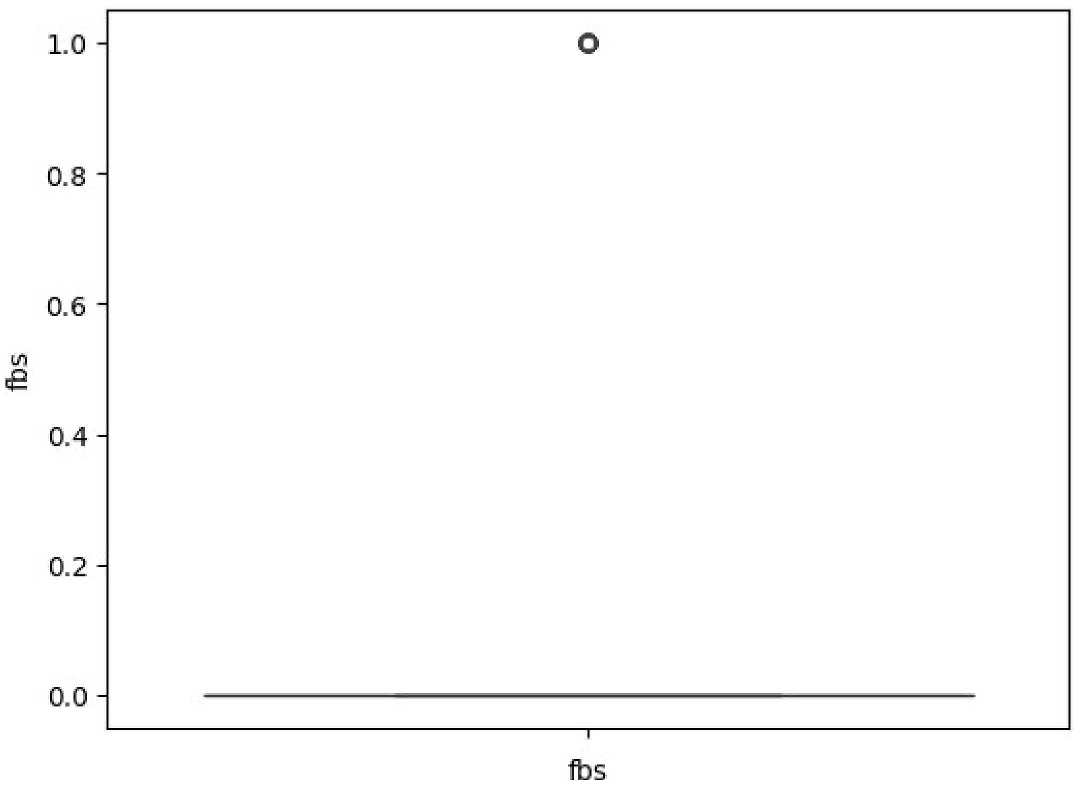

70
60
50
40
30
age
age

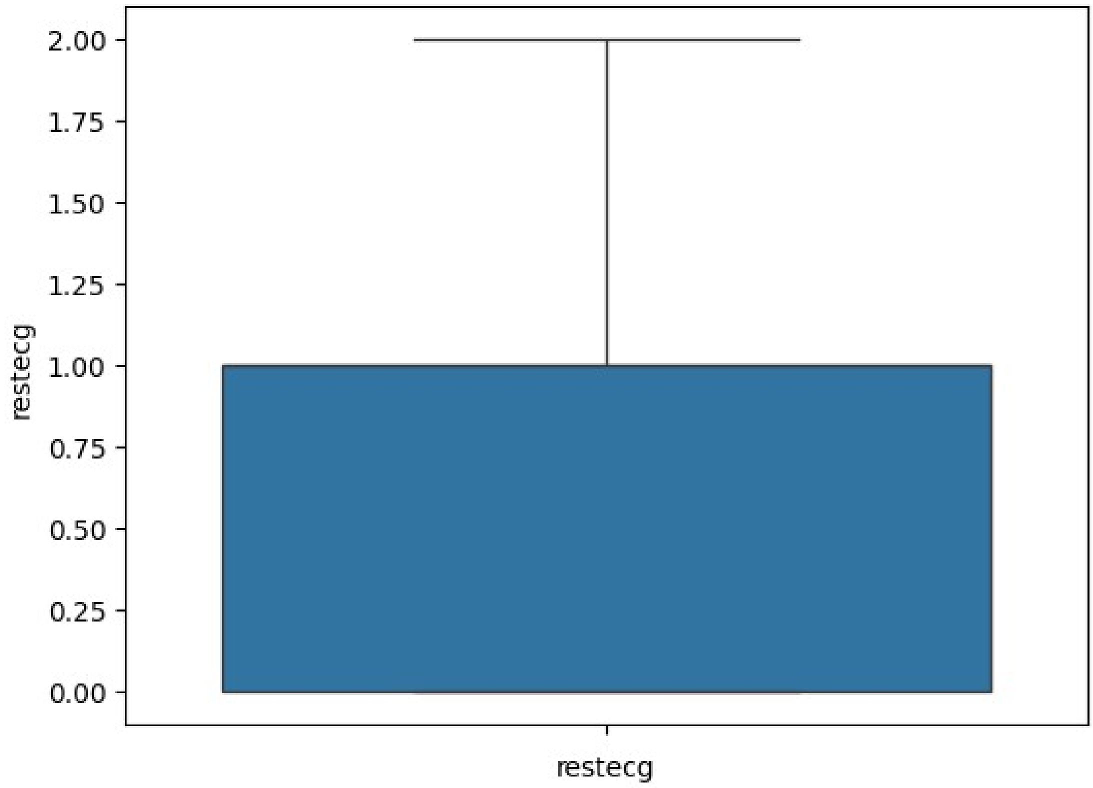

1.0
0.8
0.6
0.4
0.2
0.0
sex
sex

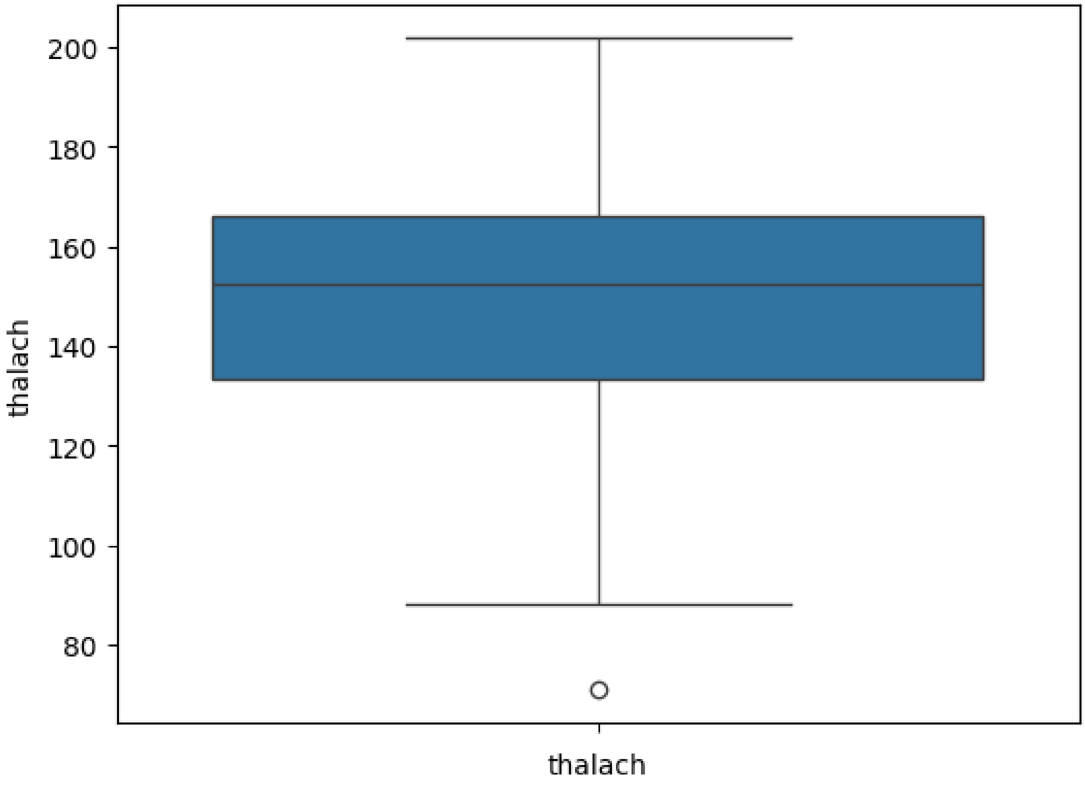

3.0
2.5
2.0
1.5
1.0
0.5
0.0
cp
cp

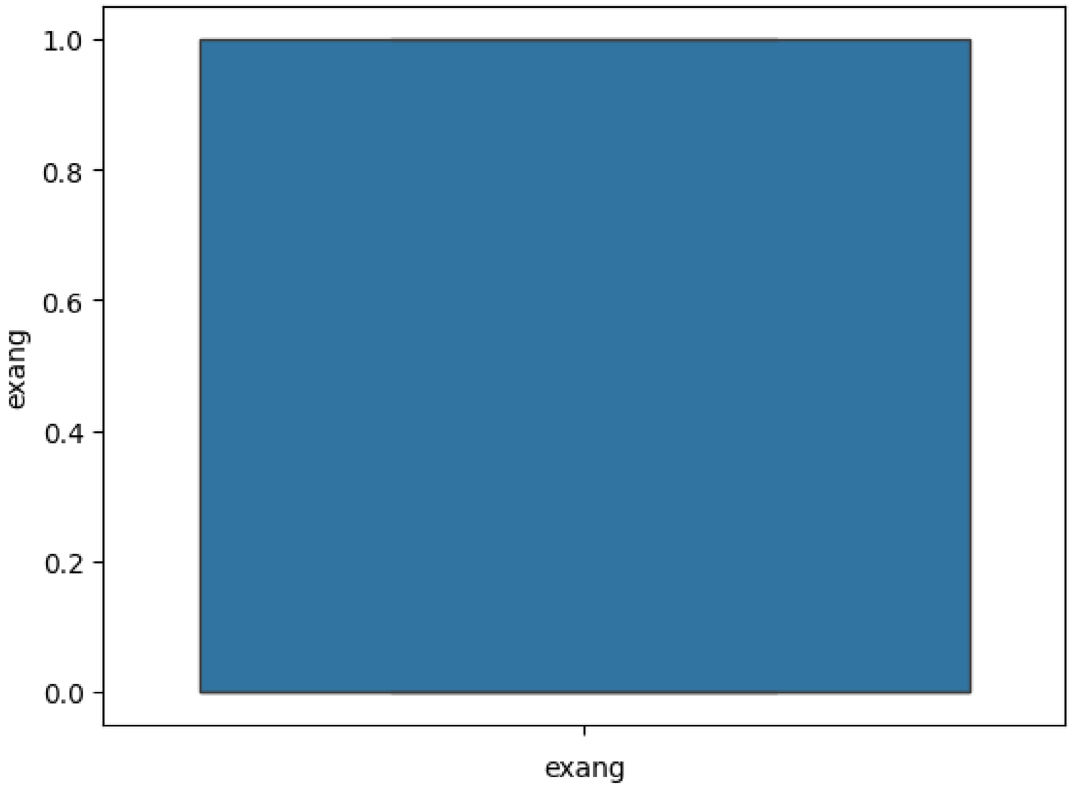

200
180
160
140
120
100
trestbps
trestbps

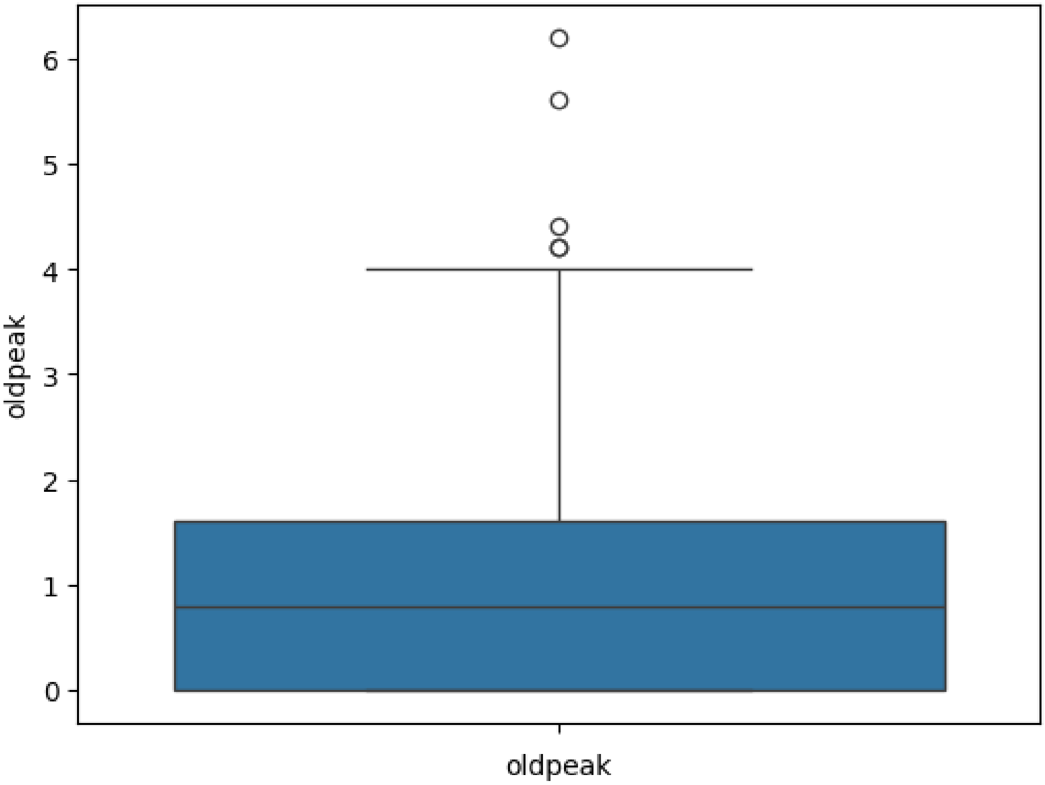

500
400
300
200
chol
chol

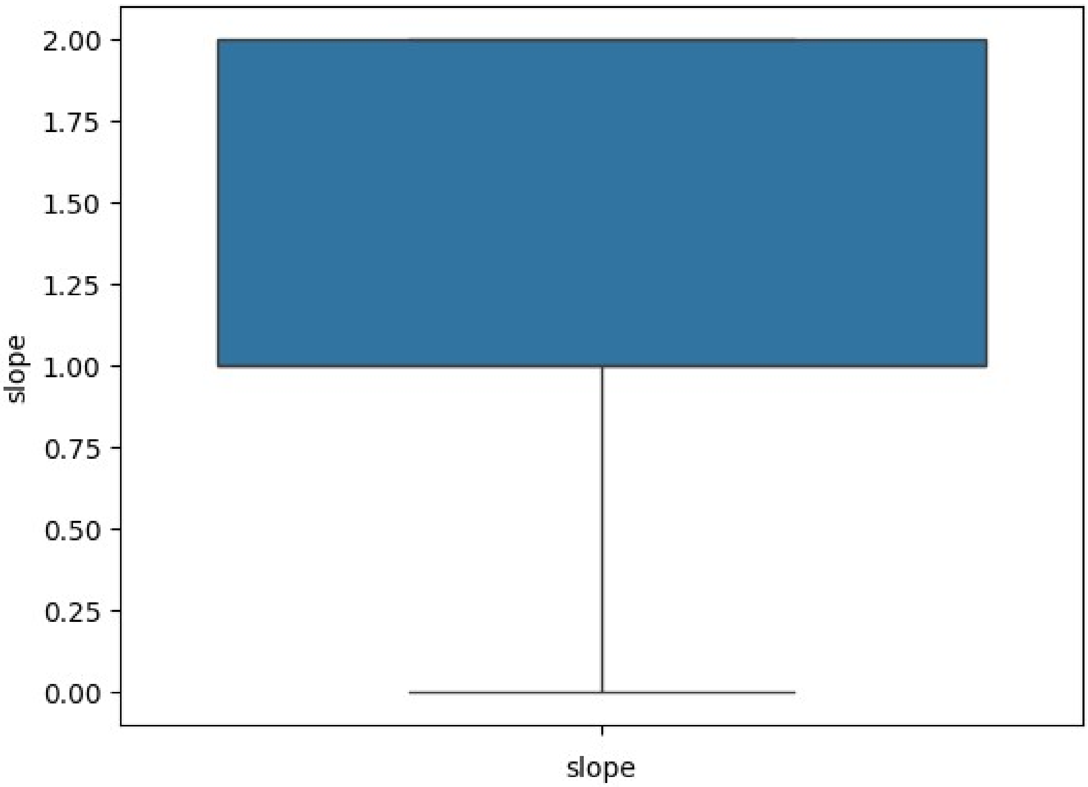

1.0
0.8
0.6
0.4
0.2
0.0
fbs
fbs

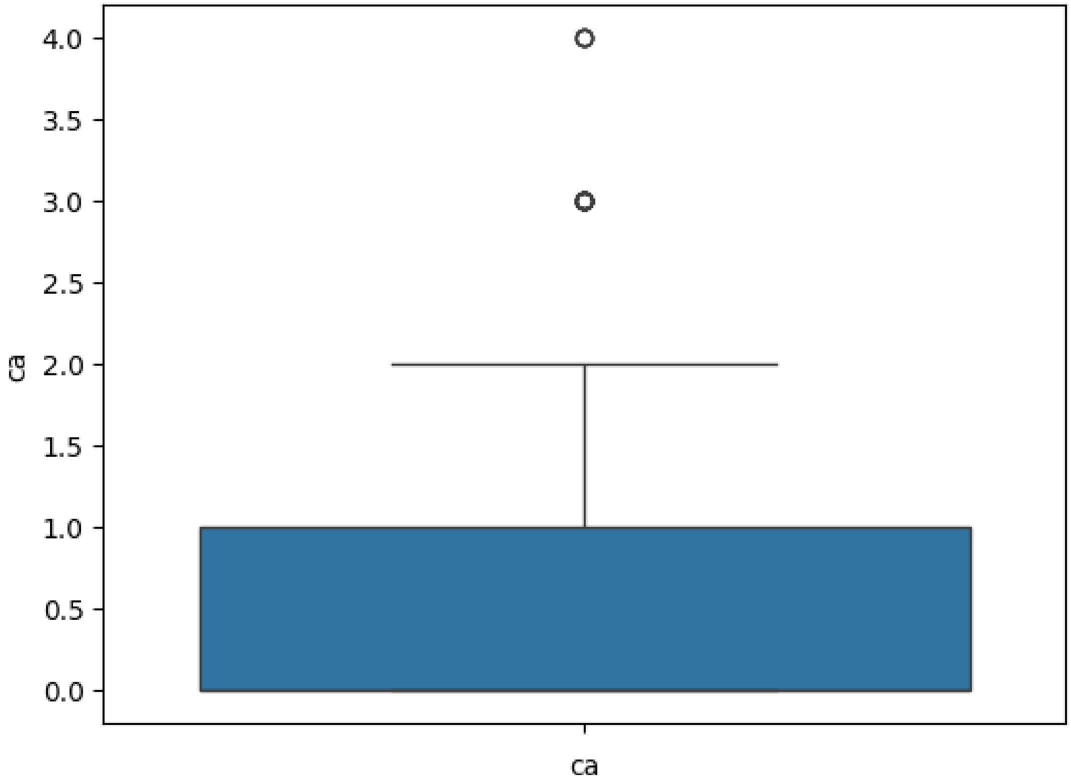

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
restecg
restecg

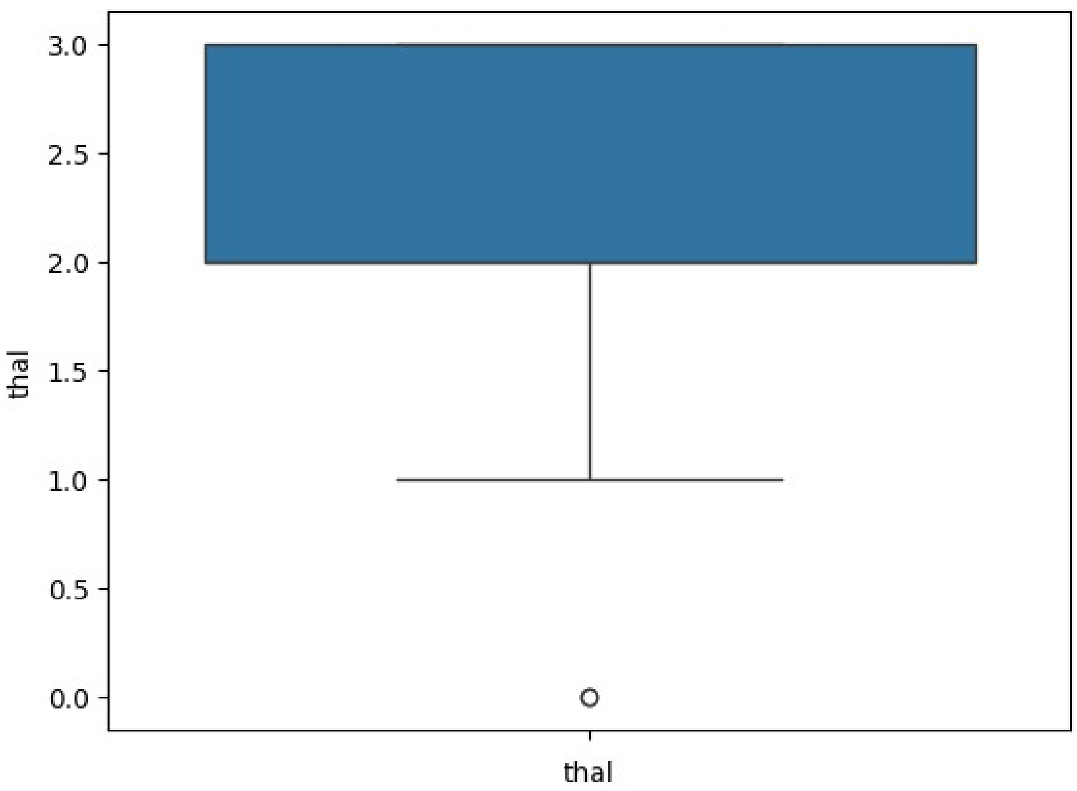

200
180
160
140
120
100
80
thalach
thalach

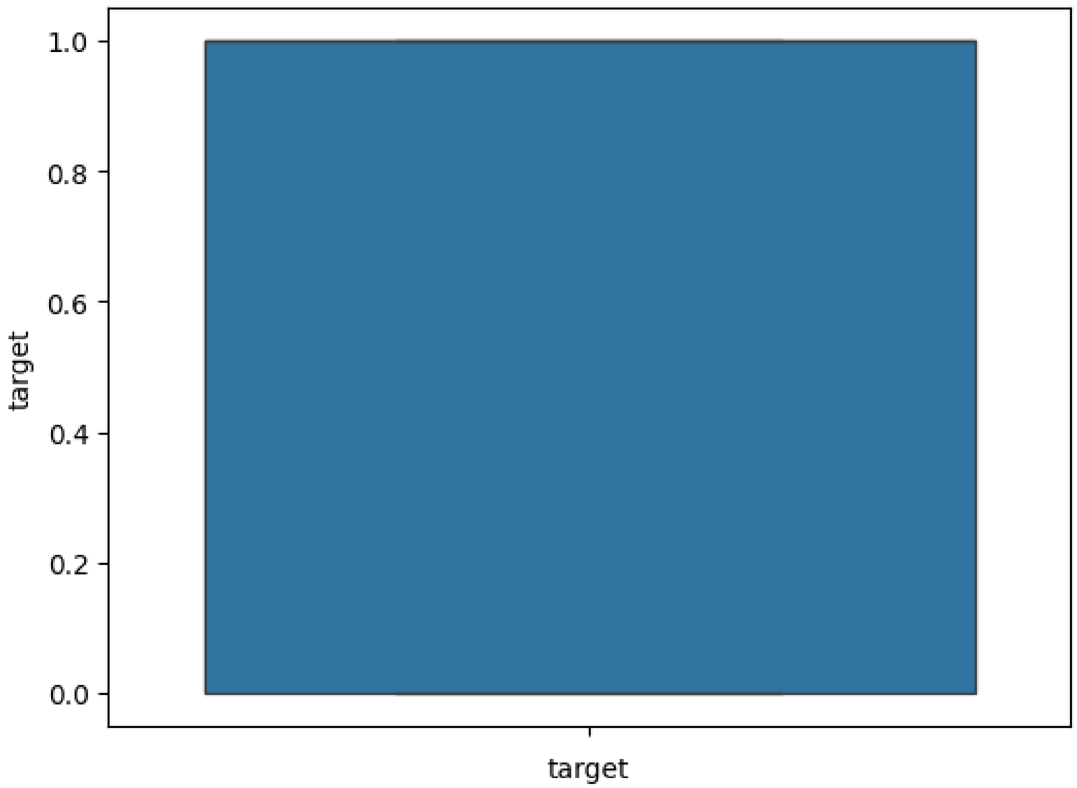

1.0
0.8
0.6
0.4
0.2
0.0
exang
exang


6
5
4
3
2
1
0
oldpeak
oldpeak


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
slope
slope


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
ca
ca


3.0
2.5
2.0
1.5
1.0
0.5
0.0
thal
thal


1.0
0.8
0.6
0.4
0.2
0.0
target
target

```
#we should not worry about ouliers in the Decision tree model as they
get ignored whil taking decision

# code for eliminating outliers

# out_col=['a','b']
# for col in out_col:
#   Q1=df[col].quantile(0.25)
#   Q3=df[col].quantile(0.75)

#   IQR=Q3-Q1

#   LB=Q1 - 1.5*(IQR)
#   UB=Q3 + 1.5*(IQR)

#   df=df[(df[col]>=LB) & (df[col]<=UB)]


#label encoding ==> no object col are there in df

# model building
# 1.split the data in terms of x and y
# 2.split in terms of train and test
# 3.model initialization
# 4.train the model
# 5.prediction by model
# 6.evaluate , accuracy
# 7.hyperparameter tuning
# 8.visualize the tree

# 1.split the data in terms of x and y

x=df.drop('target',axis=1)  # whenever we have to drop a coloumn than
we have use axis = 1 , 1 means coloumn , 0 means row
y=df['target']

x
```

```
{"summary":"{\n  \"name\": \"x\",\n  \"rows\": 302,\n  \"fields\": [\n
{\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 9,\n        \"min\": 29,\n
\"max\": 77,\n        \"num_unique_values\": 41,\n        \"samples\":
[\n          46,\n          66,\n          48\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"sex\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"cp\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 3,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          2,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"trestbps\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 17,\n        \"min\": 94,\n
\"max\": 200,\n        \"num_unique_values\": 49,\n
\"samples\": [\n          104,\n          123\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 51,\n        \"min\": 126,\n
\"max\": 564,\n        \"num_unique_values\": 152,\n
\"samples\": [\n          277,\n          169\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fbs\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"restecg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 2,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"thalach\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 22,\n        \"min\": 71,\n
\"max\": 202,\n        \"num_unique_values\": 91,\n
\"samples\": [\n          159,\n          152\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"exang\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\":
0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"oldpeak\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
1.1614522890634564,\n        \"min\": 0.0,\n        \"max\": 6.2,\n
\"num_unique_values\": 40,\n        \"samples\": [\n          1.9,\n
3.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"slope\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 2,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          0,\n
2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ca\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 0,\n        \"max\": 4,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          2,\n
4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"thal\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 3,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"x"}
```

```
y
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 302, dtype: int64
```

```
#importing the model

from sklearn.model_selection import train_test_split

# 2.split in terms of train and test

x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.20,
random_state=35)


 # test_size = 0.20 means 20% of the data points will be stored in
testing and rest 80% will go directly to the training
 #The random_state parameter in train_test_split is used to control
the random number generator that shuffles the data before splitting it
into training and testing sets

x_train
```

```
{"summary":"{\n  \"name\": \"x_train\",\n  \"rows\": 241,\n
\"fields\": [\n    {\n      \"column\": \"age\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
9,\n        \"min\": 34,\n        \"max\": 77,\n
\"num_unique_values\": 40,\n        \"samples\": [\n          42,\n
47,\n          59\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sex\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 0,\n        \"max\": 3,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"trestbps\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 16,\n        \"min\": 100,\n
\"max\": 200,\n        \"num_unique_values\": 44,\n
\"samples\": [\n          172,\n          135\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 53,\n        \"min\": 126,\n
\"max\": 564,\n        \"num_unique_values\": 137,\n
\"samples\": [\n          244,\n          318\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fbs\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"restecg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 2,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"thalach\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 22,\n        \"min\": 71,\n
\"max\": 194,\n        \"num_unique_values\": 85,\n
\"samples\": [\n          181,\n          165\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"exang\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\":
0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"oldpeak\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
1.1627978730197723,\n        \"min\": 0.0,\n        \"max\": 6.2,\n
\"num_unique_values\": 39,\n        \"samples\": [\n          1.1,\n
0.9\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"slope\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 2,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ca\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 0,\n        \"max\": 4,\n
```

```
\"num_unique_values\": 5,\n        \"samples\": [\n          4,\n
3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"thal\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 3,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          3,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"x_train"}
```

x_test

```
{"summary":"{\n  \"name\": \"x_test\",\n  \"rows\": 61,\n  \"fields\":
[\n    {\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 8,\n        \"min\": 29,\n
\"max\": 71,\n        \"num_unique_values\": 28,\n        \"samples\":
[\n          58,\n          46,\n          56\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"sex\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"cp\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 3,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          2,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"trestbps\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 21,\n        \"min\": 94,\n
\"max\": 192,\n        \"num_unique_values\": 28,\n
\"samples\": [\n          94,\n          101\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 43,\n        \"min\": 160,\n
\"max\": 342,\n        \"num_unique_values\": 52,\n
\"samples\": [\n          233,\n          197\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fbs\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"restecg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 2,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"thalach\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 22,\n        \"min\": 96,\n
\"max\": 202,\n        \"num_unique_values\": 41,\n
\"samples\": [\n          164,\n          144\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"exang\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\":
0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"oldpeak\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
1.158466104746701,\n        \"min\": 0.0,\n        \"max\": 4.2,\n
\"num_unique_values\": 22,\n        \"samples\": [\n          0.0,\n
0.9\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"slope\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 2,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          2,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ca\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 0,\n        \"max\": 4,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          4,\n
3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"thal\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 1,\n        \"max\": 3,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          2,\n
3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"x_test"}
```

```
# 3.model initialization

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()

# Training the model
model.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

```
# 5.prediction by model

y_pred=model.predict(x_test)
y_pred
```

```
array([1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1,
1,
       1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0])
```

```
y_test
```

```
22     1
92     1
17     1
121    1
113    1
      ..
27     1
117    1
124    1
194    0
279    0
Name: target, Length: 61, dtype: int64
```

```
#6.evaluate , accuracy
from sklearn.metrics import *
```

```
accuracy_score(y_test,y_pred)*100
```

```
81.9672131147541
```

```
depth=[1,2,3,4,5,6,7,8,9,10]
for i in depth:
  temp_model=DecisionTreeClassifier(max_depth=i)
  temp_model.fit(x_train,y_train)
  y_pred_temp=temp_model.predict(x_test)
  acc=accuracy_score(y_test,y_pred_temp)*100
  print(f"for the max depth {i} the accuracy score of the model is
{acc}")
```

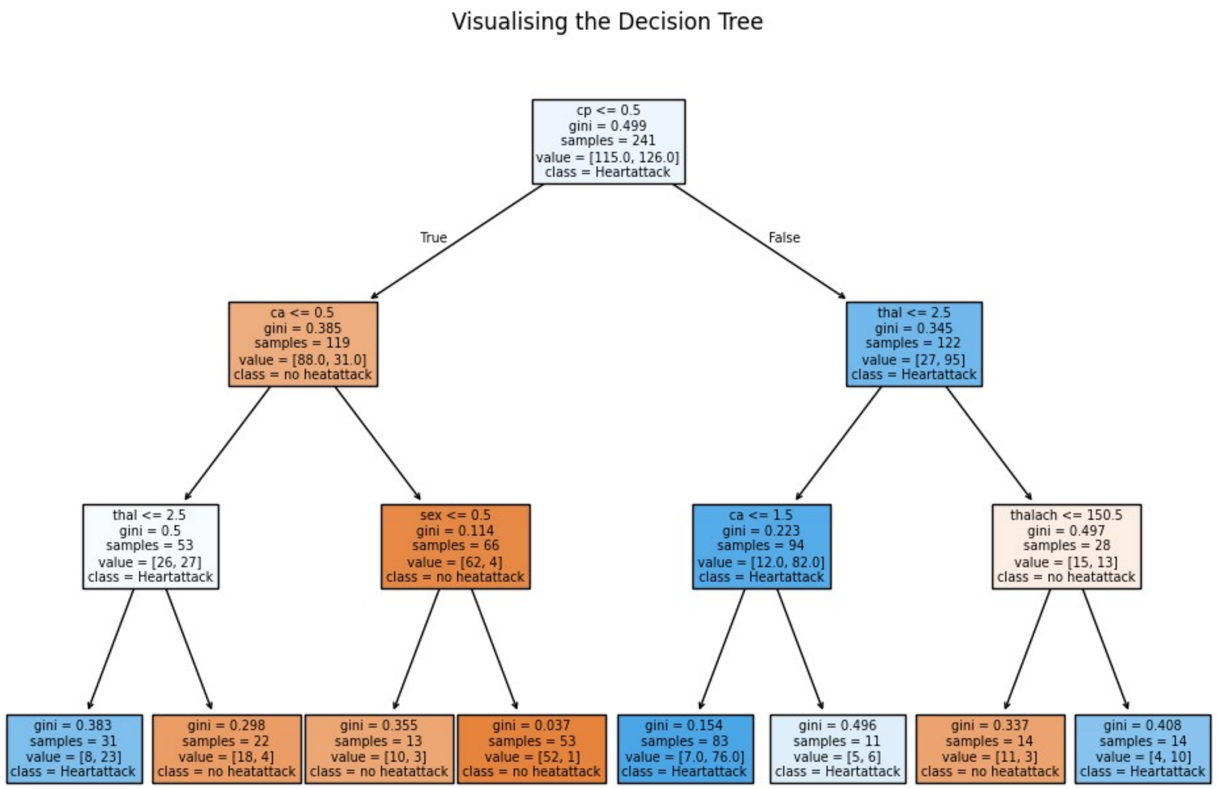
```
for the max depth 1 the accuracy score of the model is
75.40983606557377
for the max depth 2 the accuracy score of the model is
70.49180327868852
for the max depth 3 the accuracy score of the model is
80.32786885245902
for the max depth 4 the accuracy score of the model is
78.68852459016394
for the max depth 5 the accuracy score of the model is
80.32786885245902
for the max depth 6 the accuracy score of the model is
80.32786885245902
for the max depth 7 the accuracy score of the model is
78.68852459016394
for the max depth 8 the accuracy score of the model is
```

```
73.77049180327869
for the max depth 9 the accuracy score of the model is
73.77049180327869
for the max depth 10 the accuracy score of the model is
77.04918032786885
```

```
# final model


final_model=DecisionTreeClassifier(max_depth=3)
final_model.fit(x_train,y_train)
y_pred_final=final_model.predict(x_test)

accuracy_score(y_test,y_pred_final)*100
```

```
80.32786885245902
```

```
# 8.visualize the tree

from sklearn.tree import plot_tree

plt.figure(figsize=(12,8))
plot_tree(final_model, filled=True, feature_names=x.columns,
class_names=['no heatattack','Heartattack'])
plt.title('Visualising the Decision Tree')
plt.show()
```



```
#using logistic regression and determining its accuracy

from sklearn.linear_model import LogisticRegression

LR=LogisticRegression()

LR.fit(x_test,y_test)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/
_logistic.py:465: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

LogisticRegression()
```

```
y_pred_LR=LR.predict(x_test)
```

```
accuracy_score(y_test,y_pred_LR)*100
```

```
85.24590163934425
```

```
print(classification_report(y_test,y_pred_LR))
```

```
              precision    recall  f1-score   support

           0       0.85      0.74      0.79        23
           1       0.85      0.92      0.89        38

    accuracy                           0.85        61
   macro avg       0.85      0.83      0.84        61
weighted avg       0.85      0.85      0.85        61
```