b-2-heart

March 4, 2025

[1]:
```
# import pandas library
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.linear_model import LogisticRegression
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
# Reading csv file
df = pd.read_csv("Heart.csv")
df.head()
```

[2]:

	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

0.1 Data Cleaning

[3]:
```
df = df.drop_duplicates()
```

[4]:
```
# Count ,min,max ,etc of each column
df.describe()
```

[4]:

	age	sex	cp	trtbps	chol	fbs
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000

	restecg	thalachh	exng	oldpeak	slp	caa
count	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000
mean	0.526490	149.569536	0.327815	1.043046	1.397351	0.718543
std	0.526027	22.903527	0.470196	1.161452	0.616274	1.006748
min	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	133.250000	0.000000	0.000000	1.000000	0.000000
50%	1.000000	152.500000	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

	thall	output
count	302.000000	302.000000
mean	2.314570	0.543046
std	0.613026	0.498970
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

[5]:
```
# Information about each column data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 302 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       302 non-null    int64
 1   sex       302 non-null    int64
 2   cp        302 non-null    int64
 3   trtbps    302 non-null    int64
 4   chol      302 non-null    int64
 5   fbs       302 non-null    int64
 6   restecg   302 non-null    int64
 7   thalachh  302 non-null    int64
 8   exng      302 non-null    int64
```

```
 9   oldpeak   302 non-null    float64
 10  slp       302 non-null    int64
 11  caa       302 non-null    int64
 12  thall     302 non-null    int64
 13  output    302 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 35.4 KB
```

[6]:
```
#Finding null values in each column
df.isna().sum()
```

[6]:
```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

0.2 Data Integration

[7]:
```
df.head()
```

[7]:

	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

[8]:
```
df.fbs.unique()
```

[8]:
```
array([1, 0], dtype=int64)
```

[9]:
```
subSet1 = df[['age','cp','chol','thalachh']]
```

[10]:
```
subSet2 = df[['exng','slp','output']]
```

[11]:
```
merged_df = subSet1.merge(right=subSet2,how='cross')
merged_df.head()
```

[11]:

	age	cp	chol	thalachh	exng	slp	output
0	63	3	233	150	0	0	1
1	63	3	233	150	0	0	1
2	63	3	233	150	0	2	1
3	63	3	233	150	0	2	1
4	63	3	233	150	1	2	1

0.3 Error Correcting

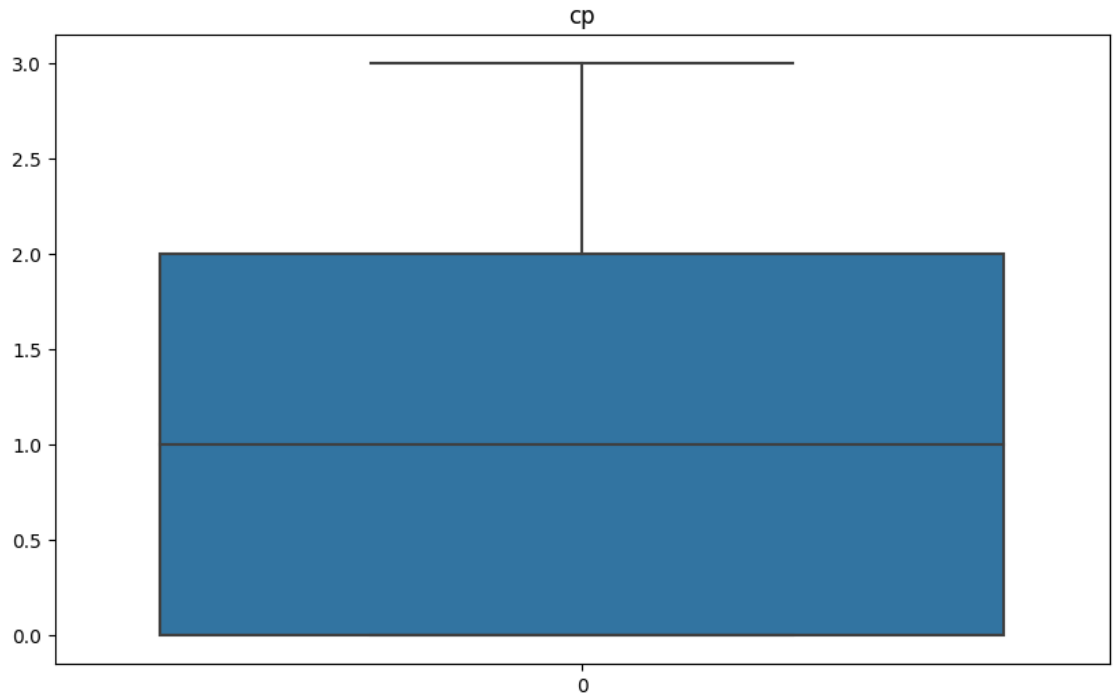
[12]:
```
df.columns
```

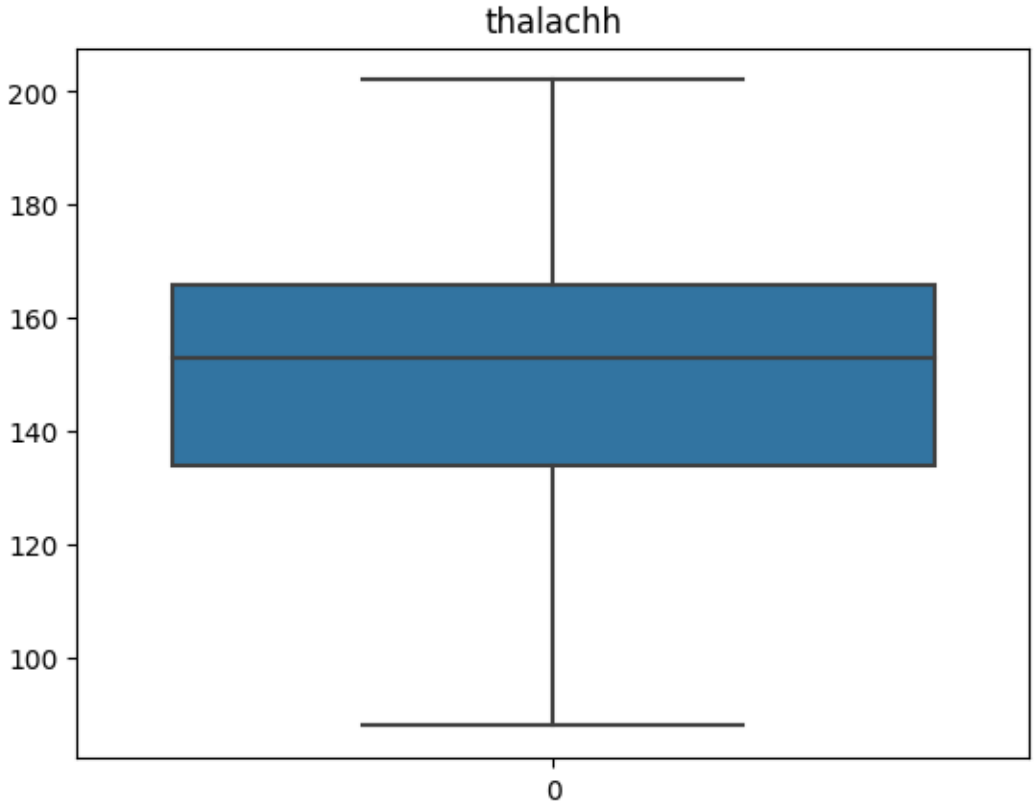
[12]:
```
Index(['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh',
       'exng', 'oldpeak', 'slp', 'caa', 'thall', 'output'],
      dtype='object')
```

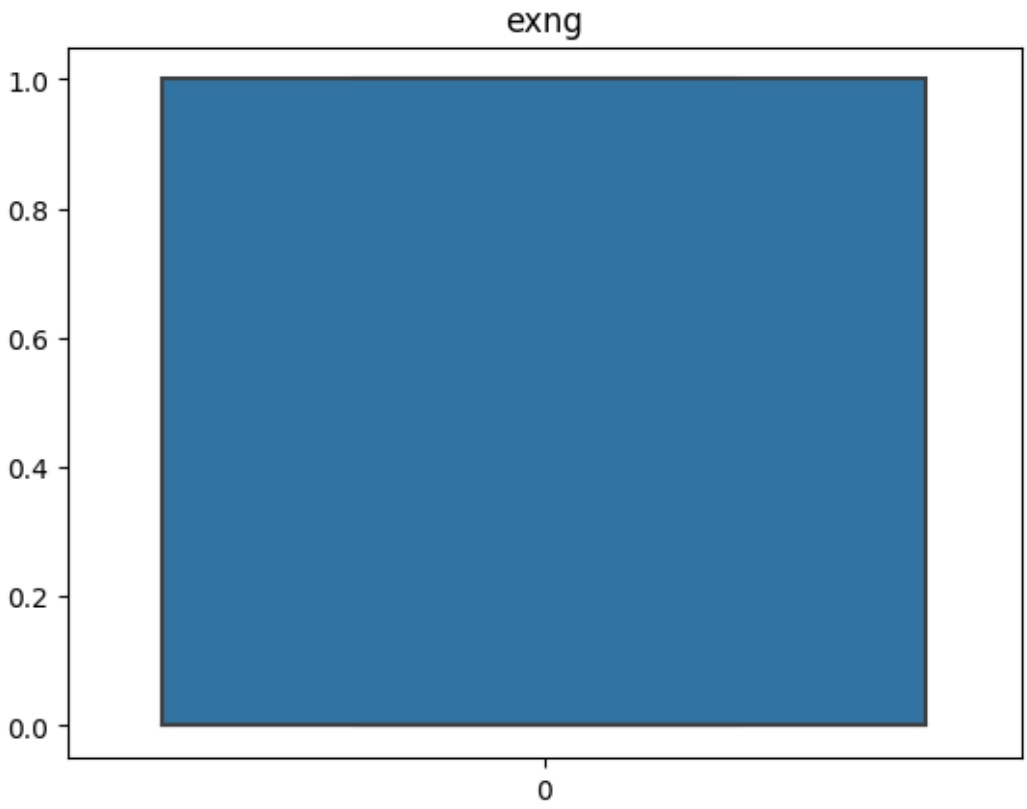
[13]:
```
def remove_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    threshold = 1.5 * IQR
    outlier_mask = (column < Q1 - threshold) | (column > Q3 + threshold)
    return column[~outlier_mask]
```

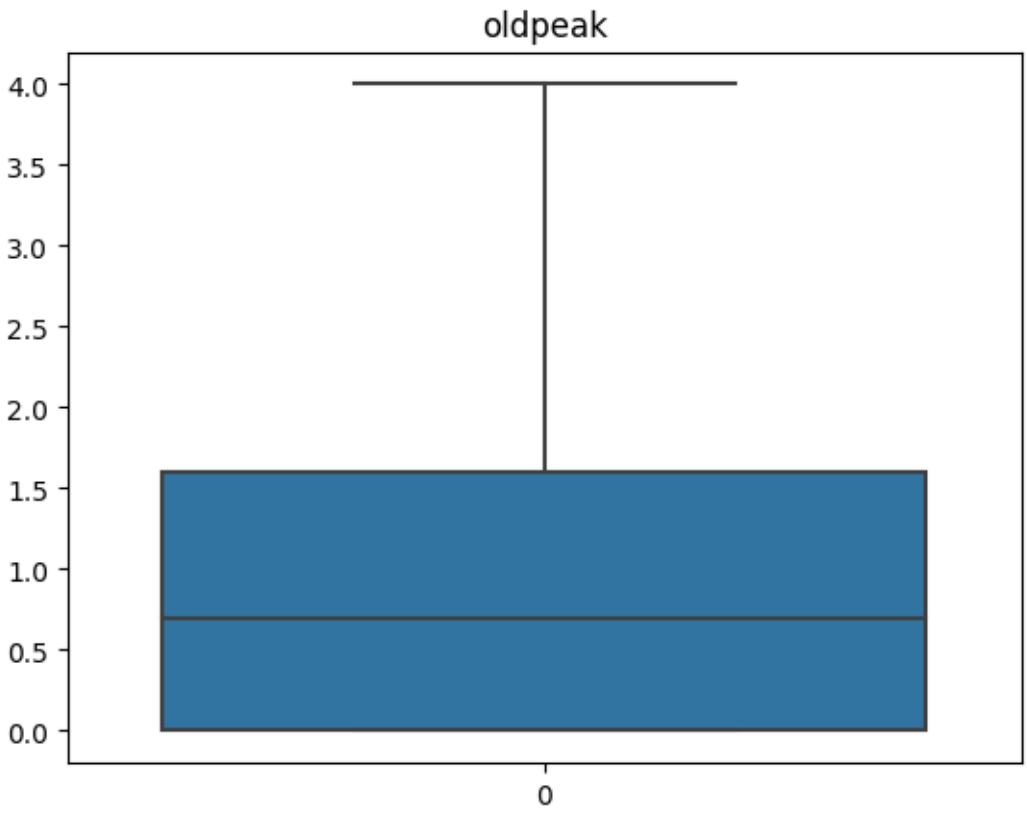
[14]:
```
# Remove outliers for each column using a loop
col_name = ['cp','thalachh','exng','oldpeak','slp','caa']
for col in col_name:
    df[col] = remove_outliers(df[col])
```

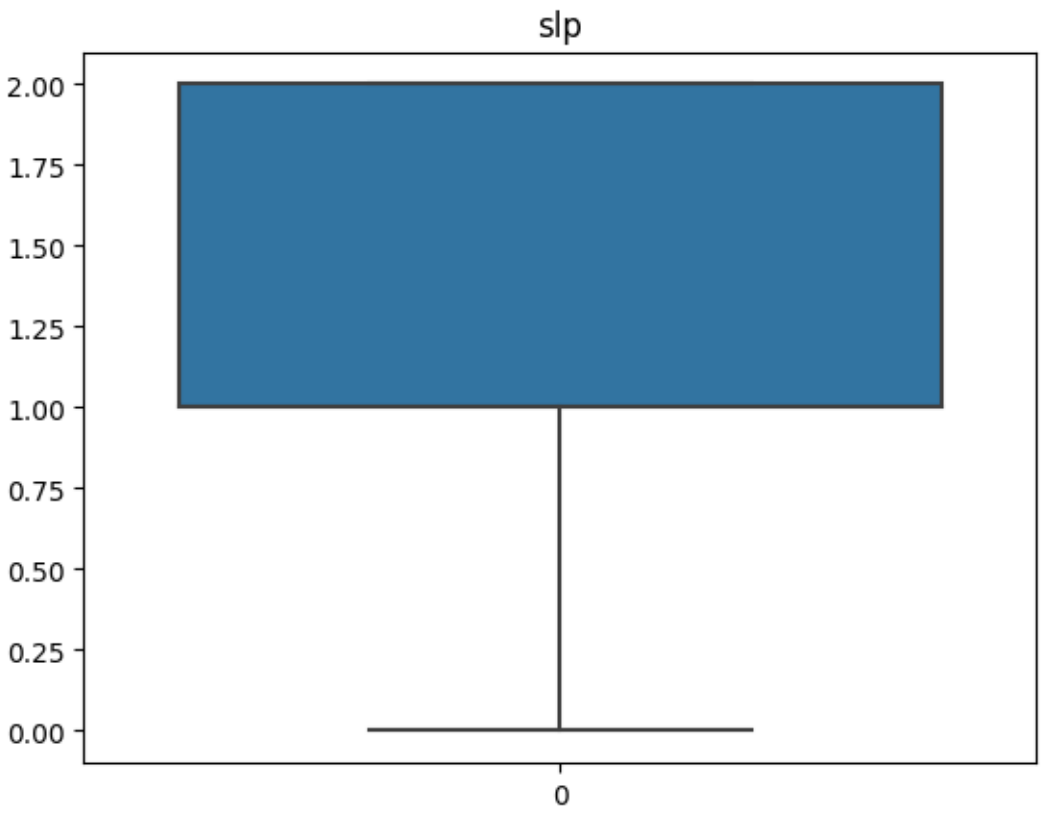
[15]:
```
plt.figure(figsize=(10, 6))  # Adjust the figure size if needed

for col in col_name:
    sns.boxplot(data=df[col])
    plt.title(col)
    plt.show()
```

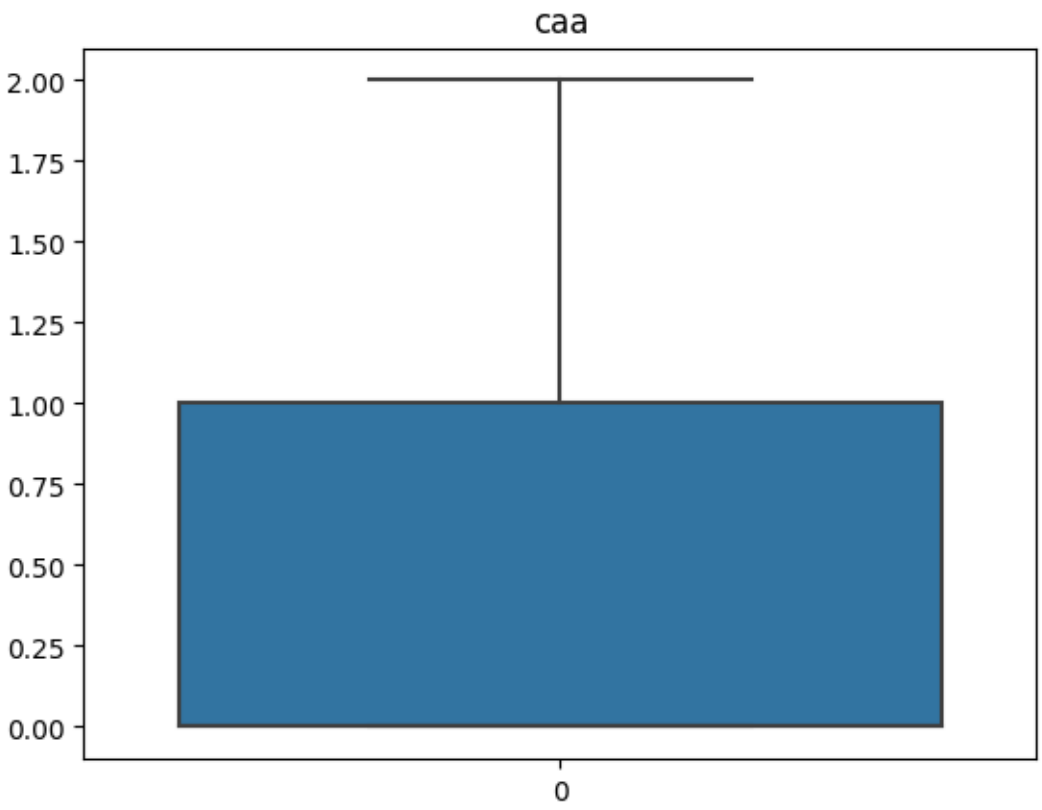

cp
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0


thalachh
200
180
160
140
120
100
0


exng
1.0
0.8
0.6
0.4
0.2
0.0
0


oldpeak
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0


slp
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0



[16]:
```
df = df.dropna()
```

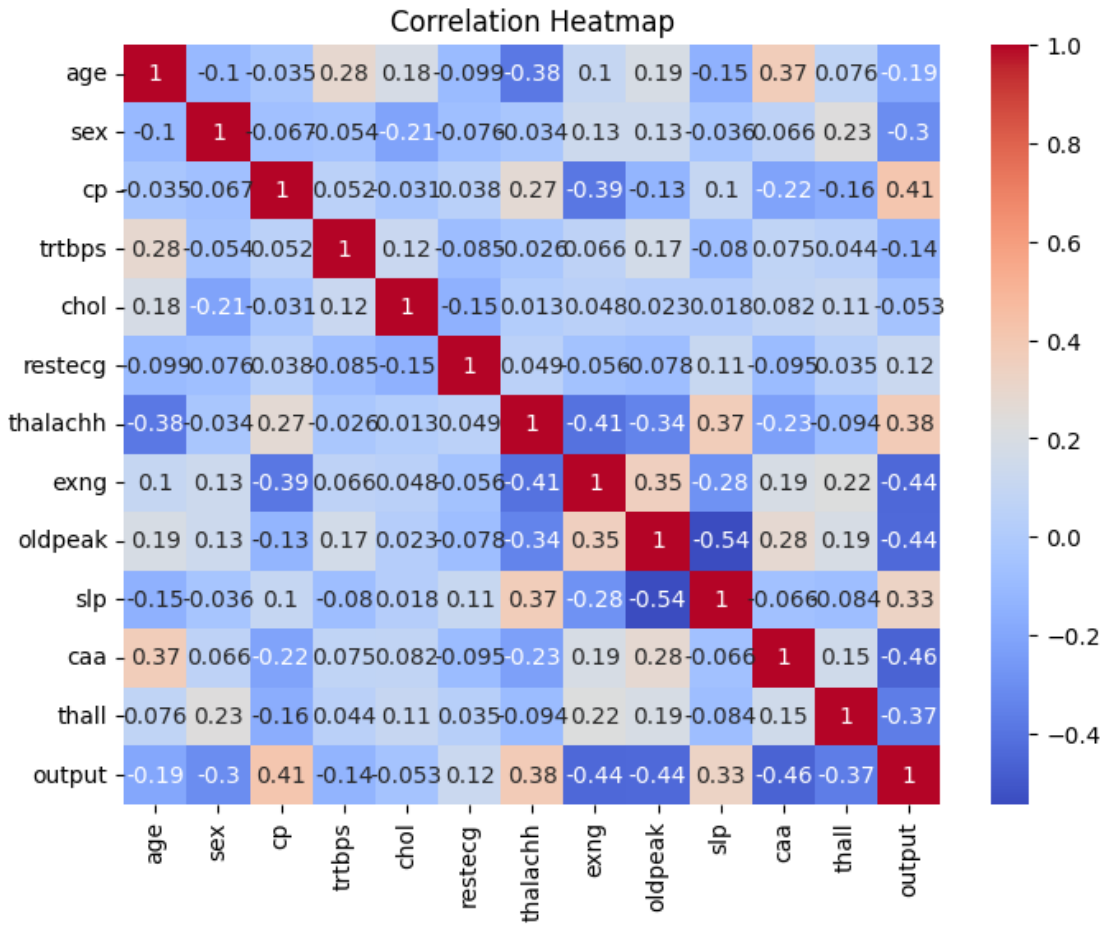
[17]:
```
df.isna().sum()
```

[17]:
```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

[18]:
```
df = df.drop('fbs',axis=1)
```

[19]:
```
# Compute correlations between features and target
correlations = df.corr()['output'].drop('output')

# Print correlations
print("Correlation with the Target:")
print(correlations)
print()

# Plot correlation heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```
Correlation with the Target:
age        -0.193798
sex        -0.303271
cp          0.410807
trtbps     -0.135238
chol       -0.052796
restecg     0.122071
thalachh    0.384609
exng       -0.444401
oldpeak    -0.437895
slp         0.329432
caa        -0.460816
thall      -0.366390
Name: output, dtype: float64
```



[20]:
```
# df.isna().sum()
```

0.4 Data Split

[21]:
```
# splitting data using train test split
x = df[['cp','thalachh','exng','oldpeak','slp','caa']]
y = df.output
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

[21]:
```
((220, 6), (55, 6), (220,), (55,))
```

0.5 Data transformation

[22]:
```
from sklearn.preprocessing import StandardScaler
```

[23]:
```
scaler = StandardScaler()
```

[24]:
```
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

[ ]:
```

```

0.6 Data model building

[25]:
```
y_train= np.array(y_train).reshape(-1, 1)
y_test= np.array(y_test).reshape(-1, 1)
```

[26]:
```
y_train.shape
```

[26]:
```
(220, 1)
```

[27]:
```
model = LogisticRegression()
model.fit(x_train_scaled, y_train)

# Make predictions on the test set
y_pred = model.predict(x_test_scaled)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8363636363636363

C:\Users\Kumbh\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\utils\validation.py:1143: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[28]:
```
#Classification model using Decision Tree
from sklearn.tree import DecisionTreeClassifier
tc=DecisionTreeClassifier(criterion='entropy')
tc.fit(x_train_scaled,y_train)
y_pred=tc.predict(x_test_scaled)

print("Training Accuracy Score :",accuracy_score(y_pred,y_test))
print("Training Confusion Matrix  :",confusion_matrix(y_pred,y_test))
```

```
Training Accuracy Score : 0.8181818181818182
Training Confusion Matrix  : [[21  4]
 [ 6 24]]
```