heart-disease-prediction-project

April 2, 2024

PROJECT TEAM'S ID:"PTID-CDS-MAR-24-1859"

PROJECT : PRCP-1016-HeartDieseasePred

INTRODUCTION

Predicting heart disease is a critical area of research and medical practice due to the significant impact cardiovascular ailments have on public health globally. Heart disease encompasses a range of conditions that affect the heart and blood vessels, including coronary artery disease, heart failure, arrhythmias, and congenital heart defects. Early detection and prediction of heart disease can greatly improve patient outcomes by enabling timely interventions and lifestyle modifications. Machine learning techniques have emerged as powerful tools in predicting heart disease risk. By analyzing various patient data such as demographic information, medical history, lifestyle factors, and clinical test results, predictive models can identify individuals at high risk of developing heart disease. These models can provide personalized risk assessments, guiding healthcare professionals in implementing preventive measures tailored to each patient's needs. In this study, we aim to explore the application of machine learning algorithms in predicting heart disease. By leveraging a dataset containing relevant patient information, we will develop and evaluate predictive models to assess the likelihood of heart disease occurrence.

PROBLEM STATEMENT

Task 1:-Prepare a complete data analysis report on the given data.

Task 2:- Create a model predicting potential Heart Diseases in people using Machine Learning algorithms.

Task3:-Suggestions to the Hospital to awake the predictions of heart diseases prevent life threats.

DATASET

1. There are 14 columns in the dataset, where the patient_id column is a unique and random identifier. The remaining 13 features are described in the section below.
2. slope_of_peak_exercise_st_segment (type: int): the slope of the peak exercise ST segment, an electrocardiography read out indicating quality of blood flow to the heart
3. thal (type: categorical): results of thallium stress test measuring blood flow to the heart, with possible values normal, fixed_defect, reversible_defect
4. resting_blood_pressure (type: int): resting blood pressure
5. chest_pain_type (type: int): chest pain type (4 values)
6. num_major_vessels (type: int): number of major vessels (0-3) colored by flourosopy
7. fasting_blood_sugar_gt_120_mg_per_dl (type: binary): fasting blood sugar > 120 mg/dl
8. resting_ekg_results (type: int): resting electrocardiographic results (values 0,1,2)
9. serum_cholesterol_mg_per_dl (type: int): serum cholestoral in mg/dl

10. oldpeak_eq_st_depression (type: float): oldpeak = ST depression induced by exercise relative to rest, a measure of abnormality in electrocardiograms
11. sex (type: binary): 0: female, 1: male
12. age (type: int): age in years
13. max_heart_rate_achieved (type: int): maximum heart rate achieved (beats per minute)
14. exercise_induced_angina (type: binary): exercise-induced chest pain (0: False, 1: True)

IMPORTING LIBRARIES

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
```

LOADING THE DATA

[ ]:
```
heart_data=pd.read_csv("/content/labels.csv")
heart_data
```

[ ]:
```
    patient_id  heart_disease_present
0       0z64un                      0
1       ryoo3j                      0
2       yt1s1x                      1
3       l2xjde                      1
4       oyt4ek                      0
..         …                      …
175     5qfar3                      1
176     2s2b1f                      1
177     nsd00i                      1
178     0xw93k                      0
179     2nx10r                      0

[180 rows x 2 columns]
```

[ ]:
```
heart_data2=pd.read_csv("/content/values.csv")
heart_data2
```

[ ]:
```
    patient_id  slope_of_peak_exercise_st_segment              thal  \
0       0z64un                                  1            normal
1       ryoo3j                                  2            normal
2       yt1s1x                                  1            normal
3       l2xjde                                  1  reversible_defect
4       oyt4ek                                  3  reversible_defect
..         …                                    …               …
```

```
175     5qfar3                                2  reversible_defect
176     2s2b1f                                2             normal
177     nsd00i                                2  reversible_defect
178     0xw93k                                1             normal
179     2nx10r                                1             normal

     resting_blood_pressure  chest_pain_type  num_major_vessels  \
0                       128                2                  0
1                       110                3                  0
2                       125                4                  3
3                       152                4                  0
4                       178                1                  0
..                      …                …                …
175                     125                4                  2
176                     180                4                  0
177                     125                3                  0
178                     124                3                  2
179                     160                3                  1

     fasting_blood_sugar_gt_120_mg_per_dl  resting_ekg_results  \
0                                       0                    2
1                                       0                    0
2                                       0                    2
3                                       0                    0
4                                       0                    2
..                                    …                    …
175                                     1                    0
176                                     0                    1
177                                     0                    0
178                                     1                    0
179                                     0                    0

     serum_cholesterol_mg_per_dl  oldpeak_eq_st_depression  sex  age  \
0                            308                       0.0    1   45
1                            214                       1.6    0   54
2                            304                       0.0    1   77
3                            223                       0.0    1   40
4                            270                       4.2    1   59
..                           …                       …  …  …
175                          254                       0.2    1   67
176                          327                       3.4    0   55
177                          309                       1.8    1   64
178                          255                       0.0    1   48
179                          201                       0.0    0   54

     max_heart_rate_achieved  exercise_induced_angina
0                        170                        0
```

```
1                        158                        0
2                        162                        1
3                        181                        0
4                        145                        0
..                       …                        …
175                      163                        0
176                      117                        1
177                      131                        1
178                      175                        0
179                      163                        0

[180 rows x 14 columns]
```

[ ]:
```
heart_data
```

[ ]:
```
    patient_id  heart_disease_present
0       0z64un                      0
1       ryoo3j                      0
2       yt1s1x                      1
3       l2xjde                      1
4       oyt4ek                      0
..         …                      …
175     5qfar3                      1
176     2s2b1f                      1
177     nsd00i                      1
178     0xw93k                      0
179     2nx10r                      0

[180 rows x 2 columns]
```

[ ]:
```
data1=pd.DataFrame(heart_data)
data1
```

[ ]:
```
    patient_id  heart_disease_present
0       0z64un                      0
1       ryoo3j                      0
2       yt1s1x                      1
3       l2xjde                      1
4       oyt4ek                      0
..         …                      …
175     5qfar3                      1
176     2s2b1f                      1
177     nsd00i                      1
178     0xw93k                      0
179     2nx10r                      0

[180 rows x 2 columns]
```

[ ]:
```
data2=pd.DataFrame(heart_data2)
data2
```

[ ]:

	patient_id	slope_of_peak_exercise_st_segment	thal
0	0z64un	1	normal
1	ryoo3j	2	normal
2	yt1s1x	1	normal
3	l2xjde	1	reversible_defect
4	oyt4ek	3	reversible_defect
..	…	…	…
175	5qfar3	2	reversible_defect
176	2s2b1f	2	normal
177	nsd00i	2	reversible_defect
178	0xw93k	1	normal
179	2nx10r	1	normal

	resting_blood_pressure	chest_pain_type	num_major_vessels
0	128	2	0
1	110	3	0
2	125	4	3
3	152	4	0
4	178	1	0
..	…	…	…
175	125	4	2
176	180	4	0
177	125	3	0
178	124	3	2
179	160	3	1

	fasting_blood_sugar_gt_120_mg_per_dl	resting_ekg_results
0	0	2
1	0	0
2	0	2
3	0	0
4	0	2
..	…	…
175	1	0
176	0	1
177	0	0
178	1	0
179	0	0

	serum_cholesterol_mg_per_dl	oldpeak_eq_st_depression	sex	age
0	308	0.0	1	45
1	214	1.6	0	54
2	304	0.0	1	77
3	223	0.0	1	40

```
4                            270                        4.2    1   59
..                           …                        …  …   …
175                          254                        0.2    1   67
176                          327                        3.4    0   55
177                          309                        1.8    1   64
178                          255                        0.0    1   48
179                          201                        0.0    0   54

     max_heart_rate_achieved  exercise_induced_angina
0                        170                        0
1                        158                        0
2                        162                        1
3                        181                        0
4                        145                        0
..                       …                        …
175                      163                        0
176                      117                        1
177                      131                        1
178                      175                        0
179                      163                        0

[180 rows x 14 columns]
```

MERGING TWO DATASETS

[ ]:
```
df=pd.merge(data1,data2,on="patient_id",how="inner")
df
```

[ ]:
```
    patient_id  heart_disease_present  slope_of_peak_exercise_st_segment  \
0       0z64un                      0                                  1
1       ryoo3j                      0                                  2
2       yt1s1x                      1                                  1
3       l2xjde                      1                                  1
4       oyt4ek                      0                                  3
..         …                      …                                  …
175     5qfar3                      1                                  2
176     2s2b1f                      1                                  2
177     nsd00i                      1                                  2
178     0xw93k                      0                                  1
179     2nx10r                      0                                  1

                  thal  resting_blood_pressure  chest_pain_type  \
0               normal                     128                2
1               normal                     110                3
2               normal                     125                4
3    reversible_defect                     152                4
4    reversible_defect                     178                1
```

```
..                 …                       …                 …
175  reversible_defect                     125                 4
176             normal                     180                 4
177  reversible_defect                     125                 3
178             normal                     124                 3
179             normal                     160                 3

     num_major_vessels  fasting_blood_sugar_gt_120_mg_per_dl  \
0                    0                                      0
1                    0                                      0
2                    3                                      0
3                    0                                      0
4                    0                                      0
..                 …                                      …
175                  2                                      1
176                  0                                      0
177                  0                                      0
178                  2                                      1
179                  1                                      0

     resting_ekg_results  serum_cholesterol_mg_per_dl  \
0                      2                          308
1                      0                          214
2                      2                          304
3                      0                          223
4                      2                          270
..                   …                            …
175                    0                          254
176                    1                          327
177                    0                          309
178                    0                          255
179                    0                          201

     oldpeak_eq_st_depression  sex  age  max_heart_rate_achieved  \
0                         0.0    1   45                      170
1                         1.6    0   54                      158
2                         0.0    1   77                      162
3                         0.0    1   40                      181
4                         4.2    1   59                      145
..                        …  …   …                            …
175                       0.2    1   67                      163
176                       3.4    0   55                      117
177                       1.8    1   64                      131
178                       0.0    1   48                      175
179                       0.0    0   54                      163

     exercise_induced_angina
```

```
0                          0
1                          0
2                          1
3                          0
4                          0
..                       …
175                        0
176                        1
177                        1
178                        0
179                        0

[180 rows x 15 columns]
```

```
[ ]: df.rename(columns={'heart_disease_present':'Outcome'},inplace=True)
```

```
[ ]: df.rename(columns={'slope_of_peak_exercise_st_segment':
     ↪'st_segment'},inplace=True)
```

```
[ ]: df.rename(columns={'resting_blood_pressure':'blood_pressure'},inplace=True)
```

```
[ ]: df.rename(columns={'chest_pain_type        ':'chest_pain'},inplace=True)
```

```
[ ]: df.rename(columns={'fasting_blood_sugar_gt_120_mg_per_dl':'sugar'},inplace=True)
```

```
[ ]: df.rename(columns={'resting_ekg_results':'ekg_result'},inplace=True)
```

```
[ ]: df.rename(columns={'serum_cholesterol_mg_per_dl':'cholestrol'},inplace=True)
```

```
[ ]: df.rename(columns={'oldpeak_eq_st_depression':'st_depression'},inplace=True)
```

```
[ ]: df.rename(columns={'max_heart_rate_achieved':'max_heartrate'},inplace=True)
```

```
[ ]: df.rename(columns={'exercise_induced_angina':'exercise'},inplace=True)
```

```
[ ]: df
```

```
[ ]:     patient_id  Outcome  st_segment               thal  blood_pressure  \
     0       0z64un        0           1             normal             128
     1       ryoo3j        0           2             normal             110
     2       yt1s1x        1           1             normal             125
     3       l2xjde        1           1  reversible_defect             152
     4       oyt4ek        0           3  reversible_defect             178
     ..         …        …         …                  …              …
     175     5qfar3        1           2  reversible_defect             125
     176     2s2b1f        1           2             normal             180
     177     nsd00i        1           2  reversible_defect             125
```

```
178     0xw93k        0           1              normal             124
179     2nx10r        0           1              normal             160

     chest_pain_type  num_major_vessels  sugar  ekg_result  cholestrol  \
0                  2                  0      0           2         308
1                  3                  0      0           0         214
2                  4                  3      0           2         304
3                  4                  0      0           0         223
4                  1                  0      0           2         270
..               ...                ...    ...         ...         ...
175                4                  2      1           0         254
176                4                  0      0           1         327
177                3                  0      0           0         309
178                3                  2      1           0         255
179                3                  1      0           0         201

     st_depression  sex  age  max_heartrate  exercise
0              0.0    1   45            170         0
1              1.6    0   54            158         0
2              0.0    1   77            162         1
3              0.0    1   40            181         0
4              4.2    1   59            145         0
..             ...  ...  ...            ...       ...
175            0.2    1   67            163         0
176            3.4    0   55            117         1
177            1.8    1   64            131         1
178            0.0    1   48            175         0
179            0.0    0   54            163         0

[180 rows x 15 columns]
```

[ ]:
```
df=df.drop(columns='patient_id',axis=1)
df
```

[ ]:
```
     Outcome  st_segment              thal  blood_pressure  chest_pain_type  \
0          0           1            normal             128                2
1          0           2            normal             110                3
2          1           1            normal             125                4
3          1           1  reversible_defect             152                4
4          0           3  reversible_defect             178                1
..       ...         ...               ...             ...              ...
175        1           2  reversible_defect             125                4
176        1           2            normal             180                4
177        1           2  reversible_defect             125                3
178        0           1            normal             124                3
179        0           1            normal             160                3
```

```
     num_major_vessels  sugar  ekg_result  cholestrol  st_depression  sex  \
0                    0      0           2         308            0.0    1
1                    0      0           0         214            1.6    0
2                    3      0           2         304            0.0    1
3                    0      0           0         223            0.0    1
4                    0      0           2         270            4.2    1
..                 …    …         …         …             …  …
175                  2      1           0         254            0.2    1
176                  0      0           1         327            3.4    0
177                  0      0           0         309            1.8    1
178                  2      1           0         255            0.0    1
179                  1      0           0         201            0.0    0

     age  max_heartrate  exercise
0     45            170         0
1     54            158         0
2     77            162         1
3     40            181         0
4     59            145         0
..   …             …       …
175   67            163         0
176   55            117         1
177   64            131         1
178   48            175         0
179   54            163         0

[180 rows x 14 columns]
```

[ ]:
```
df["thal"].value_counts()
```

[ ]:
```
normal               98
reversible_defect    74
fixed_defect          8
Name: thal, dtype: int64
```

Converting categorical values into numerical

[ ]:
```
cleanup={"thal":{"normal":1,"reversible_defect":2,"fixed_defect":3}}
cleanup
```

[ ]:
```
{'thal': {'normal': 1, 'reversible_defect': 2, 'fixed_defect': 3}}
```

[ ]:
```
df=df.replace(cleanup)
df
```

[ ]:
```
     Outcome  st_segment  thal  blood_pressure  chest_pain_type  \
0          0           1     1             128                2
```

```
1          0           2     1             110                3
2          1           1     1             125                4
3          1           1     2             152                4
4          0           3     2             178                1
..       …          …   …             …              …
175        1           2     2             125                4
176        1           2     1             180                4
177        1           2     2             125                3
178        0           1     1             124                3
179        0           1     1             160                3

     num_major_vessels  sugar  ekg_result  cholestrol  st_depression  sex  \
0                    0      0           2         308            0.0    1
1                    0      0           0         214            1.6    0
2                    3      0           2         304            0.0    1
3                    0      0           0         223            0.0    1
4                    0      0           2         270            4.2    1
..                 …    …         …          …           …  …
175                  2      1           0         254            0.2    1
176                  0      0           1         327            3.4    0
177                  0      0           0         309            1.8    1
178                  2      1           0         255            0.0    1
179                  1      0           0         201            0.0    0

     age  max_heartrate  exercise
0     45            170         0
1     54            158         0
2     77            162         1
3     40            181         0
4     59            145         0
..   …             …       …
175   67            163         0
176   55            117         1
177   64            131         1
178   48            175         0
179   54            163         0

[180 rows x 14 columns]
```

[ ]:
```
df.rename(columns={'exercise_induced_angina':'exercise'},inplace=True)
```

[ ]:
```
df
```

[ ]:
```
     Outcome  st_segment  thal  blood_pressure  chest_pain_type  \
0          0           1     1             128                2
1          0           2     1             110                3
2          1           1     1             125                4
```

```
3          1           1     2              152                4
4          0           3     2              178                1
..       …           …   …              …                  …
175        1           2     2              125                4
176        1           2     1              180                4
177        1           2     2              125                3
178        0           1     1              124                3
179        0           1     1              160                3

     num_major_vessels  sugar  ekg_result  cholestrol  st_depression  sex  \
0                    0      0           2         308            0.0    1
1                    0      0           0         214            1.6    0
2                    3      0           2         304            0.0    1
3                    0      0           0         223            0.0    1
4                    0      0           2         270            4.2    1
..                 …    …         …         …            …  …
175                  2      1           0         254            0.2    1
176                  0      0           1         327            3.4    0
177                  0      0           0         309            1.8    1
178                  2      1           0         255            0.0    1
179                  1      0           0         201            0.0    0

     age  max_heartrate  exercise
0     45            170         0
1     54            158         0
2     77            162         1
3     40            181         0
4     59            145         0
..   …             …       …
175   67            163         0
176   55            117         1
177   64            131         1
178   48            175         0
179   54            163         0

[180 rows x 14 columns]
```

[ ]:
```
df['sugar']=df['sugar'].round().astype(int) #converting float to int
```

[ ]:
```
df
```

[ ]:
```
     Outcome  st_segment  thal  blood_pressure  chest_pain_type  \
0          0           1     1             128                2
1          0           2     1             110                3
2          1           1     1             125                4
3          1           1     2             152                4
4          0           3     2             178                1
```

```
..       …          …   …              …                …
175        1          2     2              125                4
176        1          2     1              180                4
177        1          2     2              125                3
178        0          1     1              124                3
179        0          1     1              160                3

     num_major_vessels  sugar  ekg_result  cholestrol  st_depression  sex  \
0                    0      0           2         308            0.0    1
1                    0      0           0         214            1.6    0
2                    3      0           2         304            0.0    1
3                    0      0           0         223            0.0    1
4                    0      0           2         270            4.2    1
..                 …    …         …         …            …  …
175                  2      1           0         254            0.2    1
176                  0      0           1         327            3.4    0
177                  0      0           0         309            1.8    1
178                  2      1           0         255            0.0    1
179                  1      0           0         201            0.0    0

     age  max_heartrate  exercise
0     45            170         0
1     54            158         0
2     77            162         1
3     40            181         0
4     59            145         0
..   …            …       …
175   67            163         0
176   55            117         1
177   64            131         1
178   48            175         0
179   54            163         0

[180 rows x 14 columns]
```

Basic checks

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

[ ]:
```
   Outcome  st_segment  thal  blood_pressure  chest_pain_type  \
0        0           1     1             128                2
1        0           2     1             110                3
2        1           1     1             125                4
3        1           1     2             152                4
4        0           3     2             178                1

   num_major_vessels  sugar  ekg_result  cholestrol  st_depression  sex  age  \
```

```
0                 0      0           2         308            0.0    1   45
1                 0      0           0         214            1.6    0   54
2                 3      0           2         304            0.0    1   77
3                 0      0           0         223            0.0    1   40
4                 0      0           2         270            4.2    1   59

   max_heartrate  exercise
0            170         0
1            158         0
2            162         1
3            181         0
4            145         0
```

[ ]:
```
df.tail()
```

[ ]:
```
     Outcome  st_segment  thal  blood_pressure  chest_pain_type  \
175        1           2     2             125                4
176        1           2     1             180                4
177        1           2     2             125                3
178        0           1     1             124                3
179        0           1     1             160                3

     num_major_vessels  sugar  ekg_result  cholestrol  st_depression  sex  \
175                  2      1           0         254            0.2    1
176                  0      0           1         327            3.4    0
177                  0      0           0         309            1.8    1
178                  2      1           0         255            0.0    1
179                  1      0           0         201            0.0    0

     age  max_heartrate  exercise
175   67            163         0
176   55            117         1
177   64            131         1
178   48            175         0
179   54            163         0
```

[ ]:
```
df.isnull().sum()
```

[ ]:
```
Outcome             0
st_segment          0
thal                0
blood_pressure      0
chest_pain_type     0
num_major_vessels   0
sugar               0
ekg_result          0
cholestrol          0
```

```
st_depression        0
sex                  0
age                  0
max_heartrate        0
exercise             0
dtype: int64
```

[ ]:
```
df.duplicated().any()
```

[ ]:
```
False
```

[ ]:
```
df.shape
```

[ ]:
```
(180, 14)
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 180 entries, 0 to 179
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Outcome             180 non-null    int64
 1   st_segment          180 non-null    int64
 2   thal                180 non-null    int64
 3   blood_pressure      180 non-null    int64
 4   chest_pain_type     180 non-null    int64
 5   num_major_vessels   180 non-null    int64
 6   sugar               180 non-null    int64
 7   ekg_result          180 non-null    int64
 8   cholestrol          180 non-null    int64
 9   st_depression       180 non-null    float64
 10  sex                 180 non-null    int64
 11  age                 180 non-null    int64
 12  max_heartrate       180 non-null    int64
 13  exercise            180 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 21.1 KB
```

[ ]:
```
df.dtypes
```

[ ]:
```
Outcome              int64
st_segment           int64
thal                 int64
blood_pressure       int64
chest_pain_type      int64
num_major_vessels    int64
```

```
sugar                  int64
ekg_result             int64
cholestrol             int64
st_depression        float64
sex                    int64
age                    int64
max_heartrate          int64
exercise               int64
dtype: object
```

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

[ ]:
```
Index(['Outcome', 'st_segment', 'thal', 'blood_pressure', 'chest_pain_type',
       'num_major_vessels', 'sugar', 'ekg_result', 'cholestrol',
       'st_depression', 'sex', 'age', 'max_heartrate', 'exercise'],
      dtype='object')
```

[ ]:
```
df.describe()
```

[ ]:

	Outcome	st_segment	thal	blood_pressure	chest_pain_type
count	180.000000	180.000000	180.000000	180.000000	180.000000
mean	0.444444	1.550000	1.500000	131.311111	3.155556
std	0.498290	0.618838	0.583765	17.010443	0.938454
min	0.000000	1.000000	1.000000	94.000000	1.000000
25%	0.000000	1.000000	1.000000	120.000000	3.000000
50%	0.000000	1.000000	1.000000	130.000000	3.000000
75%	1.000000	2.000000	2.000000	140.000000	4.000000
max	1.000000	3.000000	3.000000	180.000000	4.000000

	num_major_vessels	sugar	ekg_result	cholestrol	st_depression
count	180.000000	180.000000	180.000000	180.000000	180.000000
mean	0.694444	0.161111	1.050000	249.211111	1.010000
std	0.969347	0.368659	0.998742	52.717969	1.121357
min	0.000000	0.000000	0.000000	126.000000	0.000000
25%	0.000000	0.000000	0.000000	213.750000	0.000000
50%	0.000000	0.000000	2.000000	245.500000	0.800000
75%	1.000000	0.000000	2.000000	281.250000	1.600000
max	3.000000	1.000000	2.000000	564.000000	6.200000

	sex	age	max_heartrate	exercise
count	180.000000	180.000000	180.000000	180.000000
mean	0.688889	54.811111	149.483333	0.316667
std	0.464239	9.334737	22.063513	0.466474
min	0.000000	29.000000	96.000000	0.000000
25%	0.000000	48.000000	132.000000	0.000000
50%	1.000000	55.000000	152.000000	0.000000
75%	1.000000	62.000000	166.250000	1.000000

```
max       1.000000    77.000000      202.000000     1.000000
```

[ ]:
```
df.Outcome.unique()
```

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

[ ]:
```
df.Outcome.value_counts()
```

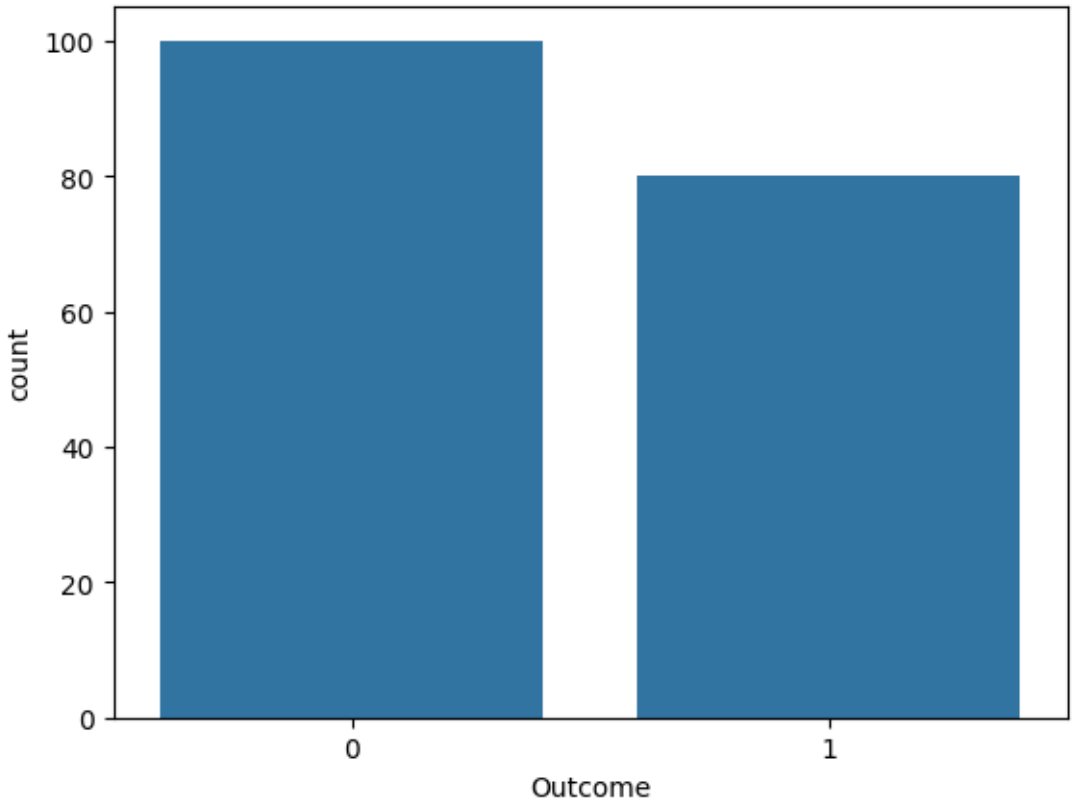
[ ]:
```
0    100
1     80
Name: Outcome, dtype: int64
```

Exploratory Data Analysis (EDA)

Univariate

[ ]:
```
sns.countplot(x="Outcome",data=df)
```

[ ]:
```
<Axes: xlabel='Outcome', ylabel='count'>
```

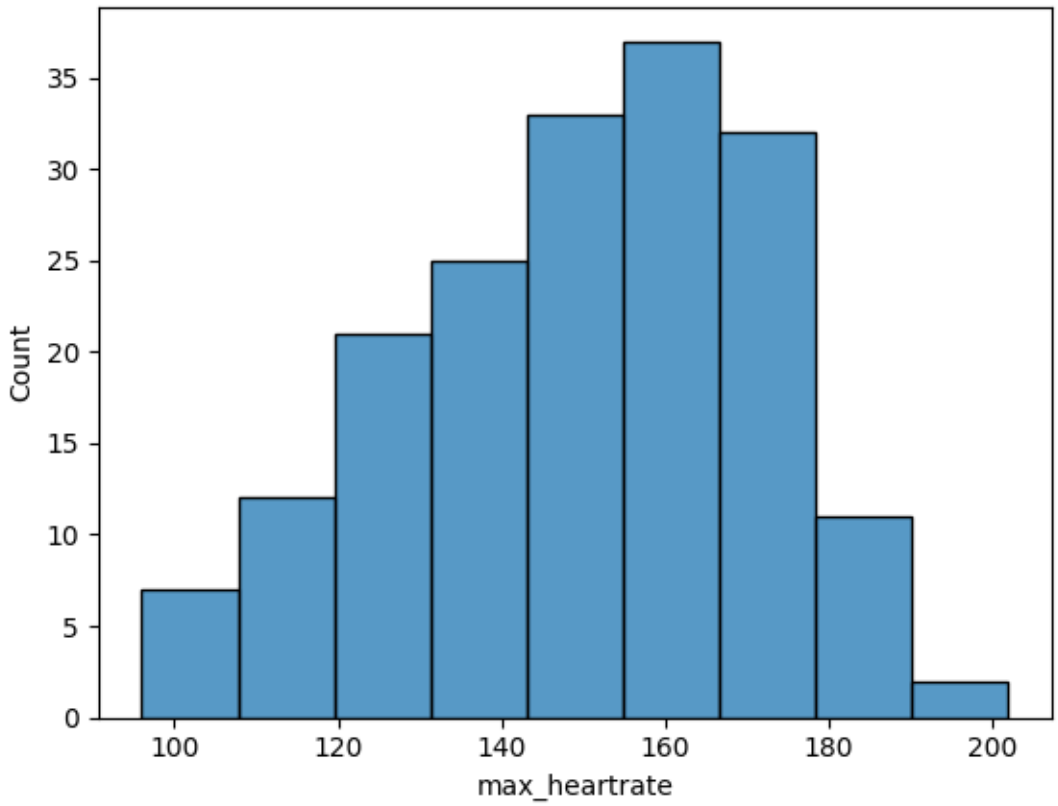


Insights: Among 180 patient records, The number of people suffering from heart disease are less then number of people who doesn't have the disease. 80 patients are suffering from Heart disease.

100 patients are are not suffering from the Heart disease.

[ ]:
```
sns.histplot(x="max_heartrate",data=df)
```

[ ]:
```
<Axes: xlabel='max_heartrate', ylabel='Count'>
```

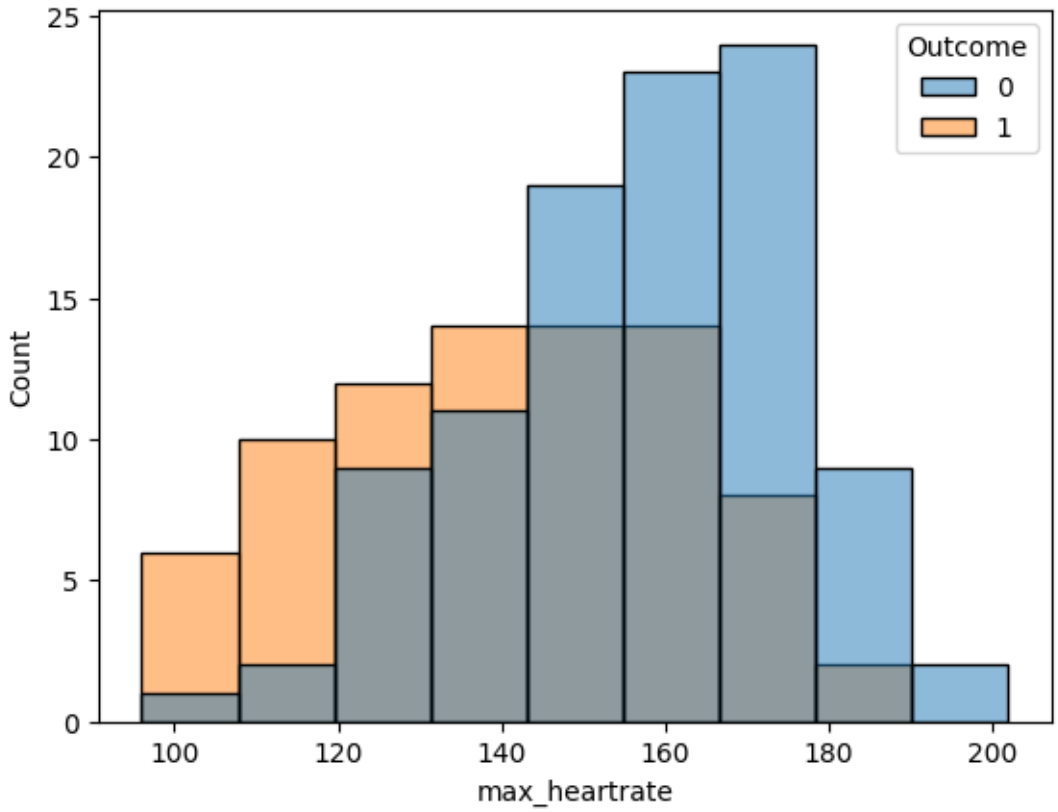


Insights: This histogram plot is showing the frequency distributions.It is showing the number of observation within each interval. Around 5 to 10 people are having maximum heartrate between 98 to 108. Around 10 to 15 people are having maximum heartrate between 108 to 119. Around 20 to 25 people are having maximum heartrate between 119 to 132. 25 people are having maximum heartrate between 132 to 142. Around 30 to 35 people are having maximum heartrate between 142 to 155. More then 35 people are having maximum heartrate between 155 to 167. Around 30 to 35 people are having maximum heartrate between 167 to 178. Around 10 to 15 people are having maximum heartrate between 178 to 191. Around 0 to 5 people are having maximum heartrate between 191 to 202.

The max_heartrate mentioned above are not the exact values.

[ ]:
```
sns.histplot(x="max_heartrate",data=df,hue="Outcome")
```

[ ]:
```
<Axes: xlabel='max_heartrate', ylabel='Count'>
```



Insights: This histogram plot is showing the frequency distributions.It is showing the number of observation within each interval.

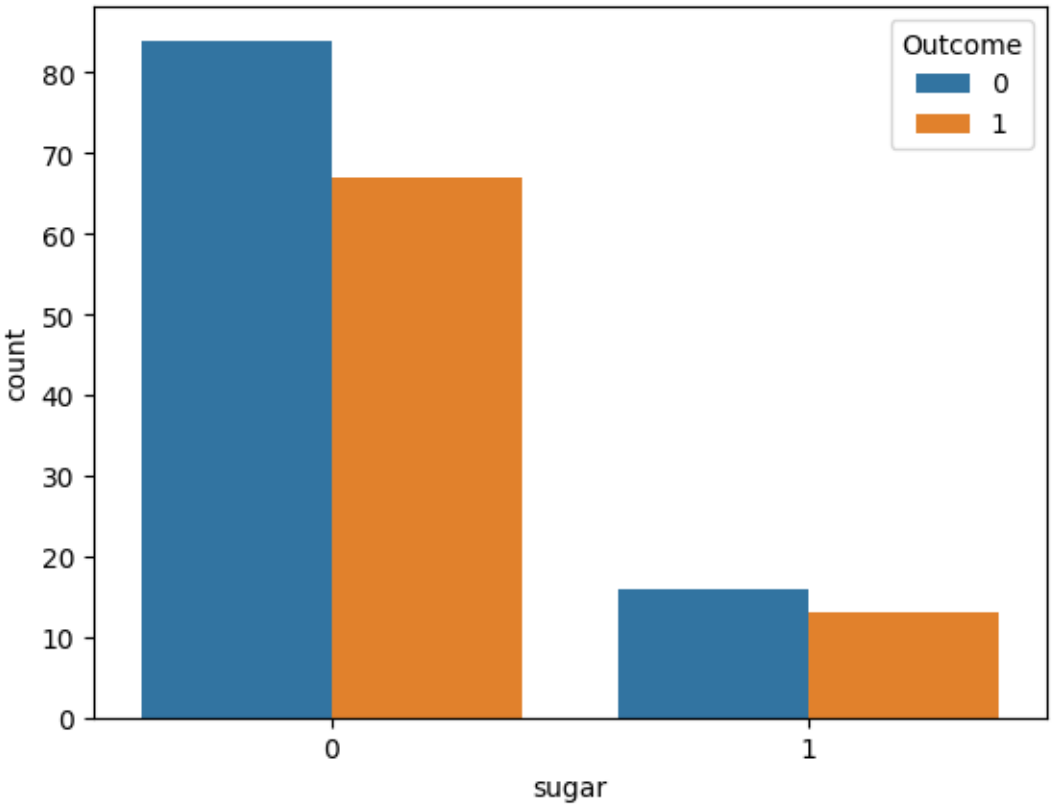
Around 5 to 10 people are having maximum heartrate between 98 to 108. Around 10 to 15 people are having maximum heartrate between 108 to 119. Around 20 to 25 people are having maximum heartrate between 119 to 132. 25 people are having maximum heartrate between 132 to 142. Around 30 to 35 people are having maximum heartrate between 142 to 155. More then 35 people are having maximum heartrate between 155 to 167. Around 30 to 35 people are having maximum heartrate between 167 to 178. Around 10 to 15 people are having maximum heartrate between 178 to 191. Around 0 to 5 people are having maximum heartrate between 191 to 202.

The max_heartrate mentioned above are not the exact values.

The hue parameter is commonly used to color-code data points based on a specific column in the dataset.The orange color shows the number of people who have the heart disease.The blue color shows the number of people who does not have heart disease.The gray color shows the combination of both.

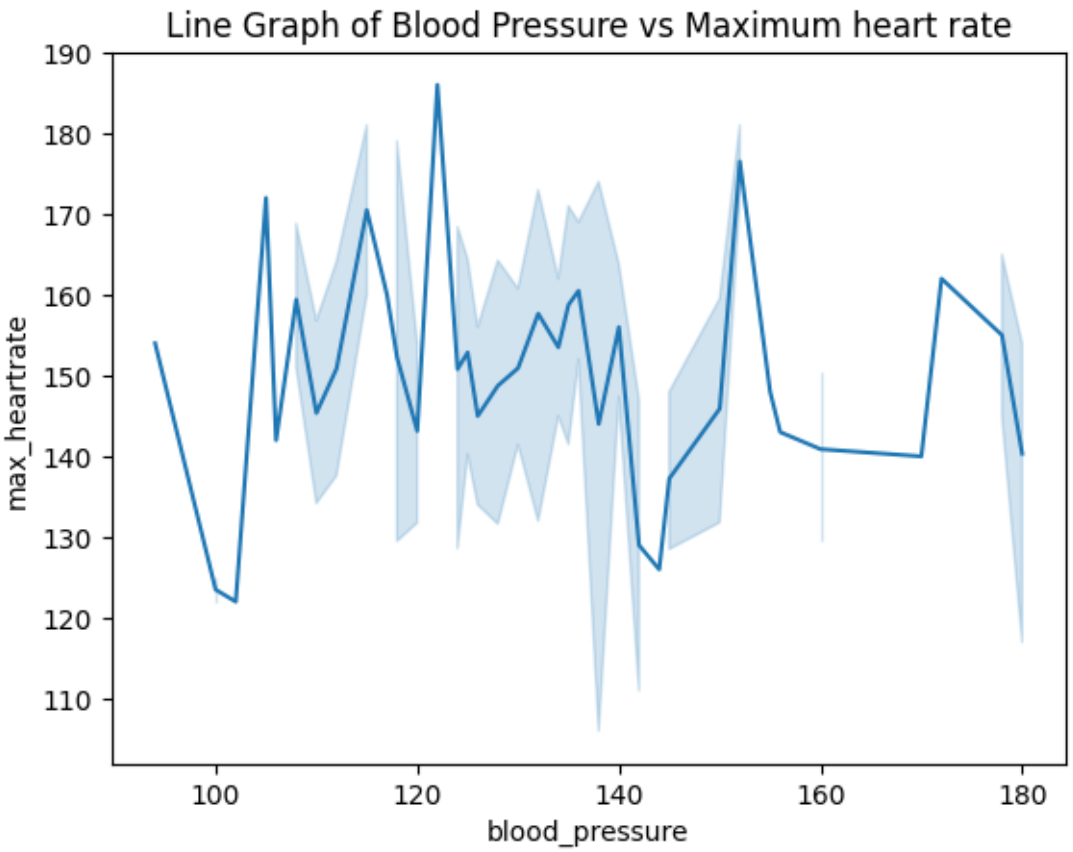
[ ]:
```
sns.countplot(x="sugar",hue="Outcome",data=df)
```

[ ]:
```
<Axes: xlabel='sugar', ylabel='count'>
```



Insights: This countplot shows us the count of sugar level of a people. The number of People who has sugar are less than the number of people who doesn't have sugar. Among the people who has sugar,less number of people has heart disease and even in the records of people not having sugar,the more number of people doesn't have heart disease.

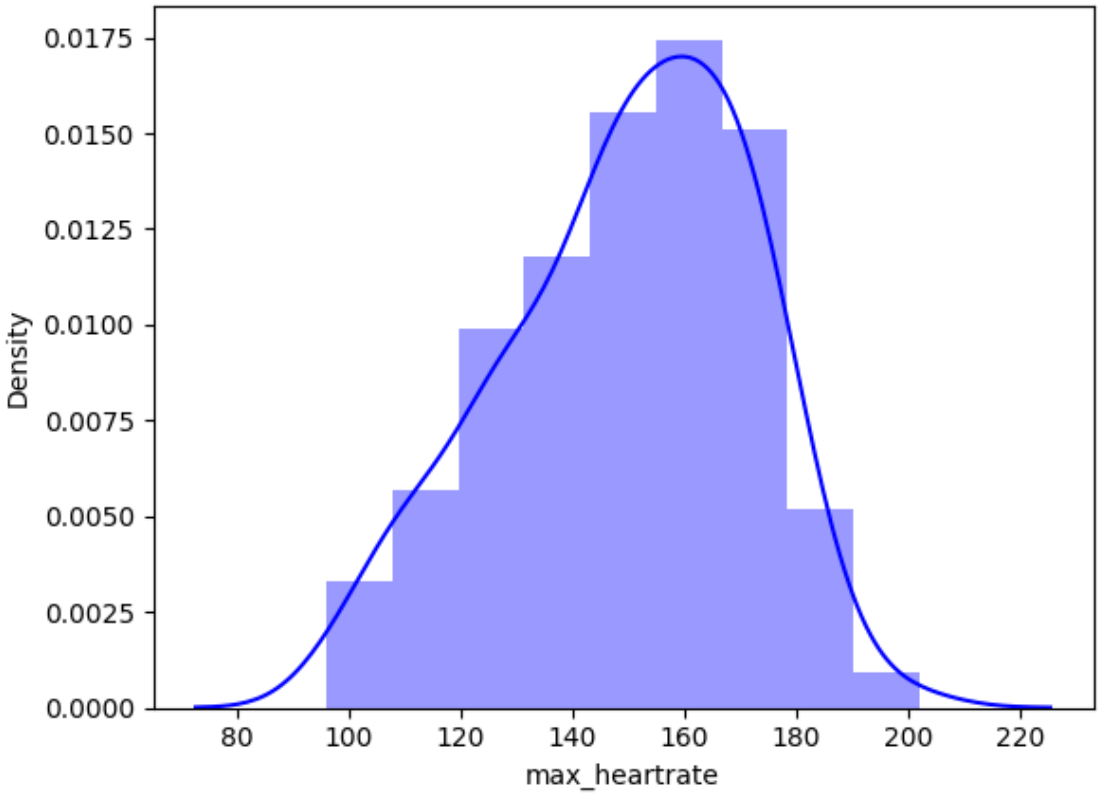
[ ]:
```
import warnings
warnings.filterwarnings("ignore")
sns.lineplot(x="blood_pressure",y="max_heartrate",data=df).set_title("Line␣
 ↪Graph of Blood Pressure vs Maximum heart rate") # if Confidence interval is␣
 ↪shaded part ie for sepal length 5 the sepal width can be range from 2.7 to 3.
 ↪3,if we dont want shaded part put ci=None
plt.show()
```



Insights: A line plot is a graphical representation of data in which individual data points are plotted along a line to display the relationship between two variables.It is showing the graph between blood pressure and maximum heart rate.

Distplot:

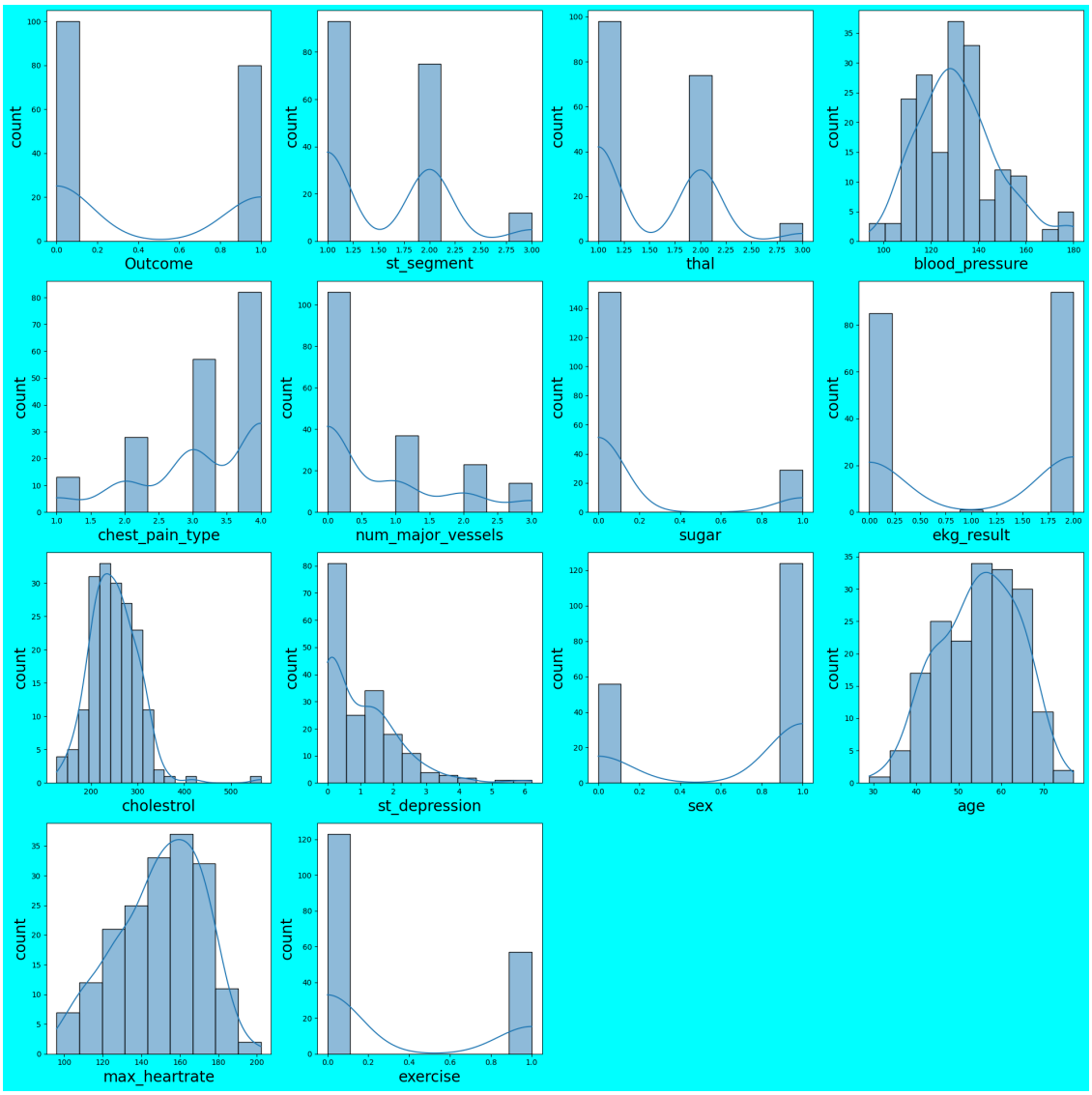
[ ]:
```
sns.distplot(df["max_heartrate"],color="b")#,hist=False)#kde=False)
plt.show()
#density means the probability of the given point to lie on the curve
```



Insights:

will be used to unvariant set of observation. It visualizes it through a histogram

[ ]:
```
plt.figure(figsize=(20,20),facecolor="aqua")
plotnumber=1
for column in df:
  if plotnumber<=14:
    ax=plt.subplot(4,4,plotnumber)
    sns.histplot(df[column],kde=True)
    plt.xlabel(column,fontsize=20)
    plt.ylabel('count',fontsize=20)
  plotnumber+=1
plt.tight_layout()
```



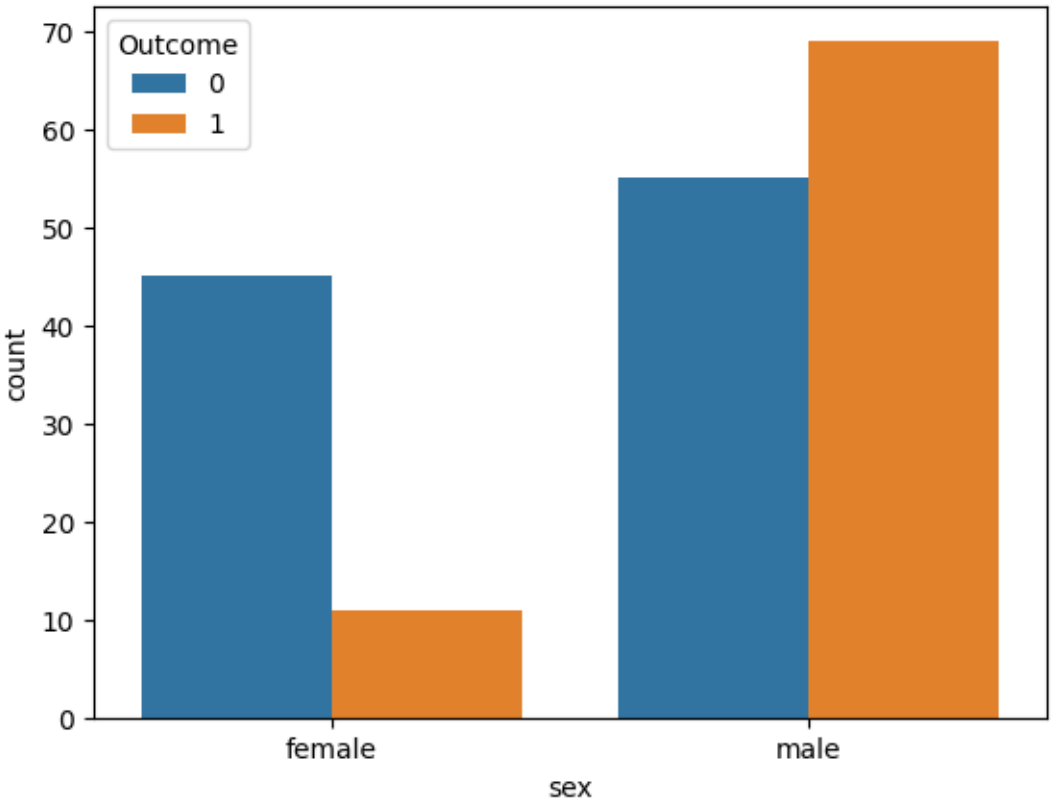
Insights:

This histogram plot is showing the frequency distributions.It is showing the number of observation within each interval.

This plot is showing the count of each column.

Bivariate

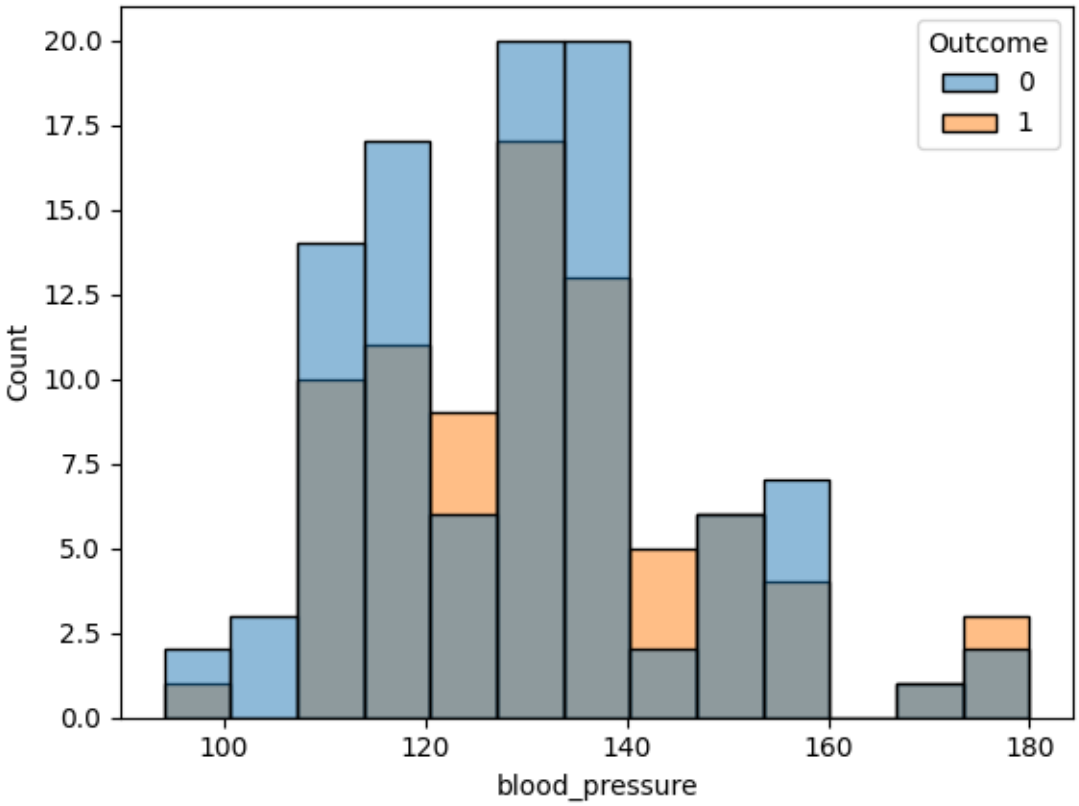
[ ]:
```
sns.countplot(x="sex",hue="Outcome",data=df)
plt.xticks([0,1],['female','male'])
plt.show()
```



Insights: This countplot shows the count of sex of people. The number of male are more compared to female. Among female around 10-20 are having heart disease and 40-50 people are not having heart disease. Among male around 60-70 are having heart disease and 50-60 people are not having heart disease

[ ]:
```
sns.histplot(x="blood_pressure",hue="Outcome",data=df)
```

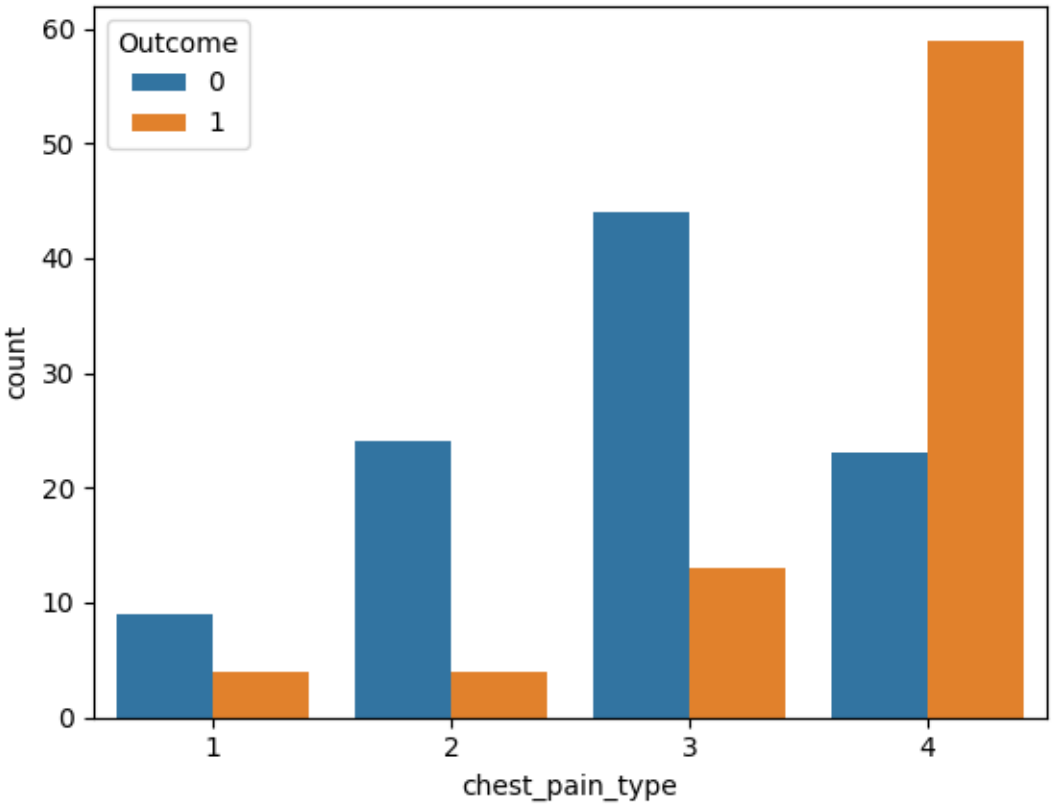
[ ]:
```
<Axes: xlabel='blood_pressure', ylabel='Count'>
```



Insights: Histplot is showing the count of blood pressure of the patients. Among 0 to 20 people highest blood pressure is between 120 to 140.

[ ]:
```
sns.countplot(x="chest_pain_type",hue="Outcome",data=df)
```

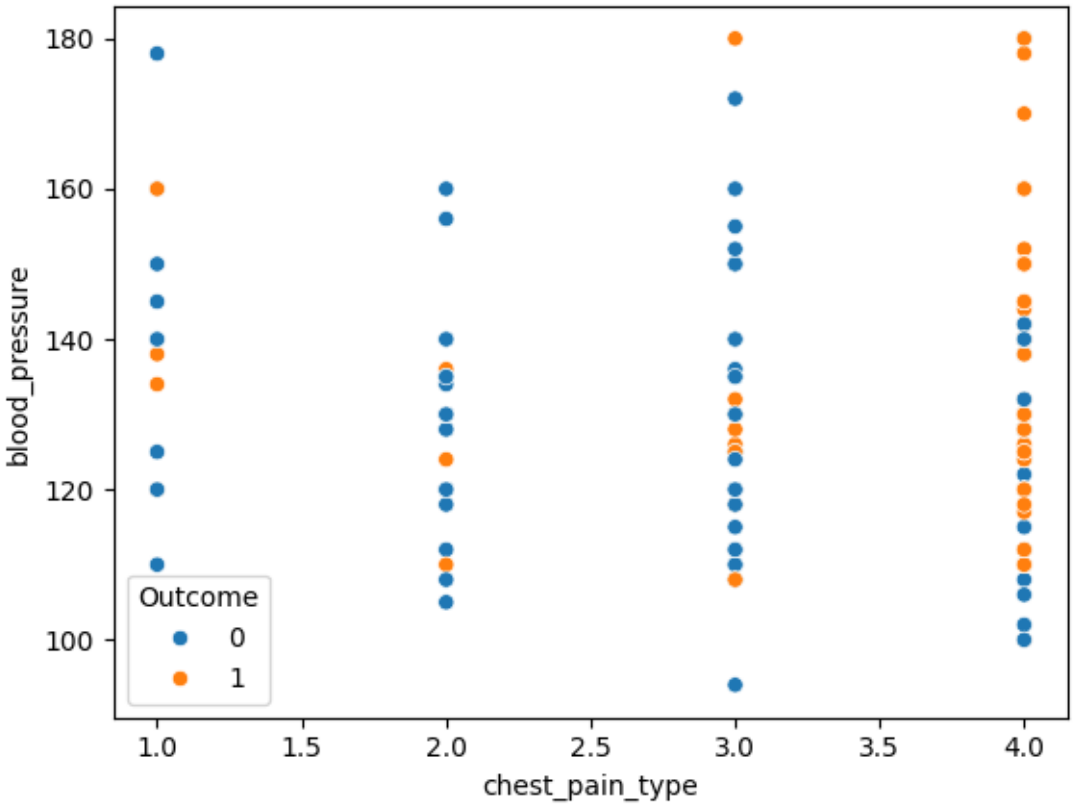
[ ]:
```
<Axes: xlabel='chest_pain_type', ylabel='count'>
```



Insights: This count plot is showing us count of people having heart disease on the basis of type of chest pain. Among the people who are having the chest pain type 1(typical angina),nearly 10 people doesn't have heart disease and nearly 5 are having heart disease. Among the people who are having the chest pain type2 (atypical angina),the people having heart disease are less than people who dont have heart disease and it is same with the chest pain type 3(non-anginal pain). Among the people who are having the chest pain type 4(asymptomatic),the number people having heart disease are greater than number of people not having heart disease.

[ ]:
```
sns.scatterplot(x="chest_pain_type",y="blood_pressure",hue="Outcome",data=df)
```

[ ]:
```
<Axes: xlabel='chest_pain_type', ylabel='blood_pressure'>
```

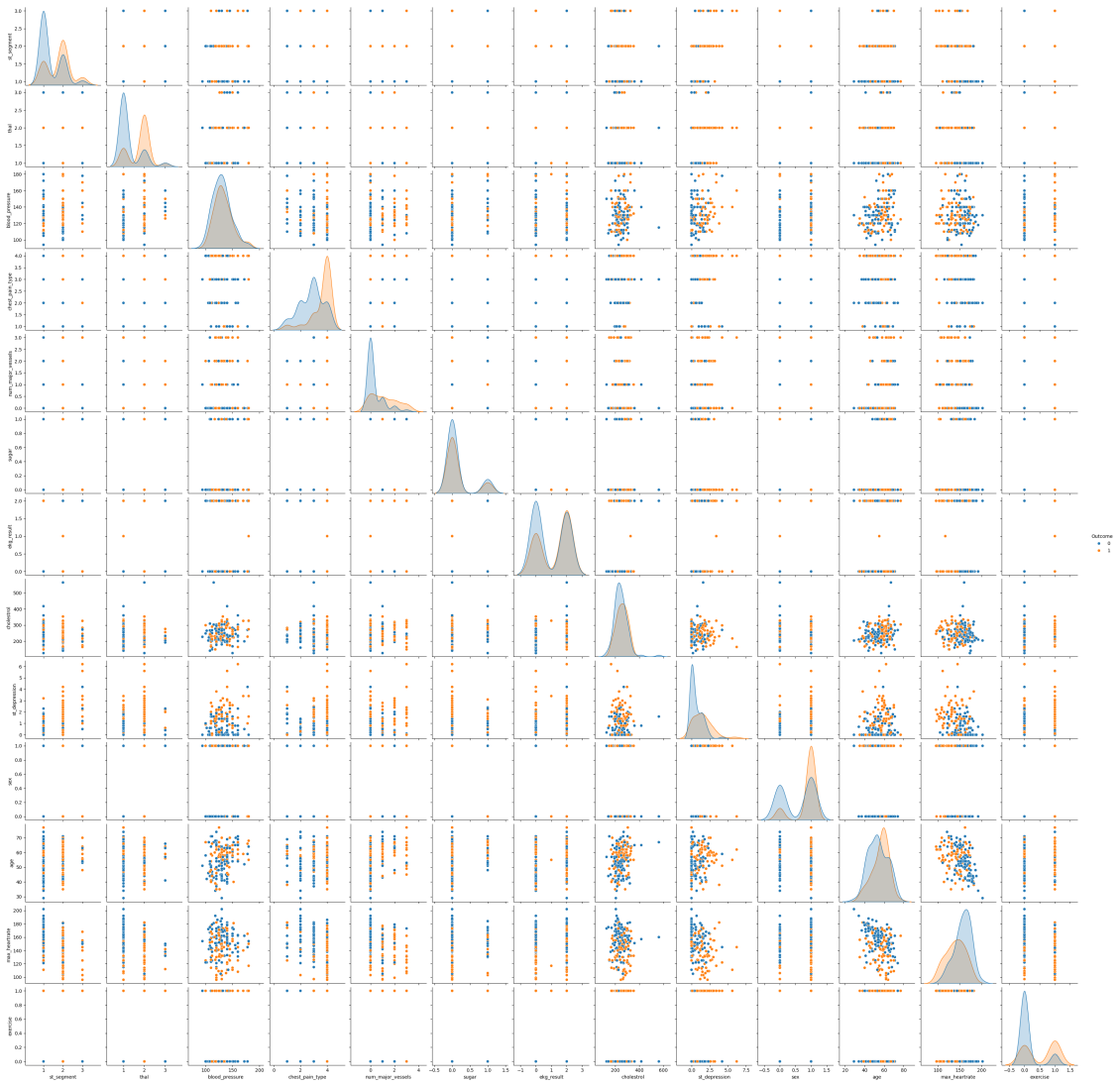


Insights: Scatter plot is a graph of two sets of data along the two axes. It is used to visualize the relationship between the two variables. In this scatterplot we have plotted chest pain type and blood pressure to see the relationship among them.

Multivariate

[ ]:
```
sns.pairplot(df,hue="Outcome")
```

[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7d46ac6ab670>
```



Insights: A pair plot, also known as a scatterplot matrix, is a matrix of graphs that enables the visualization of the relationship between each pair of variables in a dataset. This pairplot is showing the relationship of every columns with each other.

Data Preprocessing

[ ]:
```
df.isnull().sum()
```

[ ]:
```
Outcome              0
st_segment           0
thal                 0
blood_pressure       0
chest_pain_type      0
num_major_vessels    0
sugar                0
```

```
ekg_result          0
cholestrol          0
st_depression       0
sex                 0
age                 0
max_heartrate       0
exercise            0
dtype: int64
```

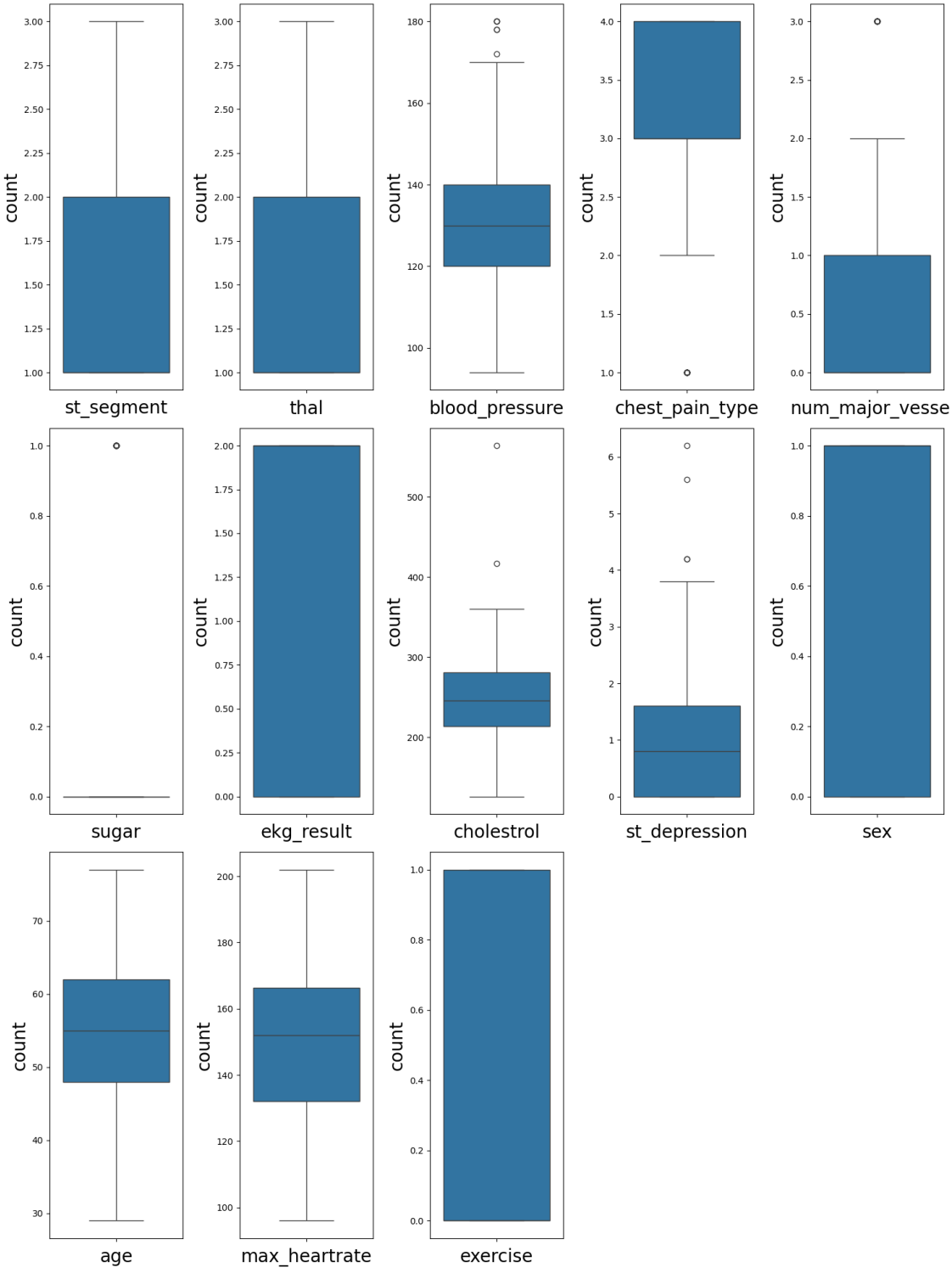
[ ]:
```
df.describe()
```

[ ]:
```
          Outcome  st_segment        thal  blood_pressure  chest_pain_type  \
count  180.000000  180.000000  180.000000      180.000000       180.000000
mean     0.444444    1.550000    1.500000      131.311111         3.155556
std      0.498290    0.618838    0.583765       17.010443         0.938454
min      0.000000    1.000000    1.000000       94.000000         1.000000
25%      0.000000    1.000000    1.000000      120.000000         3.000000
50%      0.000000    1.000000    1.000000      130.000000         3.000000
75%      1.000000    2.000000    2.000000      140.000000         4.000000
max      1.000000    3.000000    3.000000      180.000000         4.000000

       num_major_vessels       sugar  ekg_result  cholestrol  st_depression  \
count         180.000000  180.000000  180.000000  180.000000     180.000000
mean            0.694444    0.161111    1.050000  249.211111       1.010000
std             0.969347    0.368659    0.998742   52.717969       1.121357
min             0.000000    0.000000    0.000000  126.000000       0.000000
25%             0.000000    0.000000    0.000000  213.750000       0.000000
50%             0.000000    0.000000    2.000000  245.500000       0.800000
75%             1.000000    0.000000    2.000000  281.250000       1.600000
max             3.000000    1.000000    2.000000  564.000000       6.200000

              sex         age  max_heartrate    exercise
count  180.000000  180.000000     180.000000  180.000000
mean     0.688889   54.811111     149.483333    0.316667
std      0.464239    9.334737      22.063513    0.466474
min      0.000000   29.000000      96.000000    0.000000
25%      0.000000   48.000000     132.000000    0.000000
50%      1.000000   55.000000     152.000000    0.000000
75%      1.000000   62.000000     166.250000    1.000000
max      1.000000   77.000000     202.000000    1.000000
```

Box Plot: We are plotting a box plot for each column to check whether there are outliers and handle them accordingly.

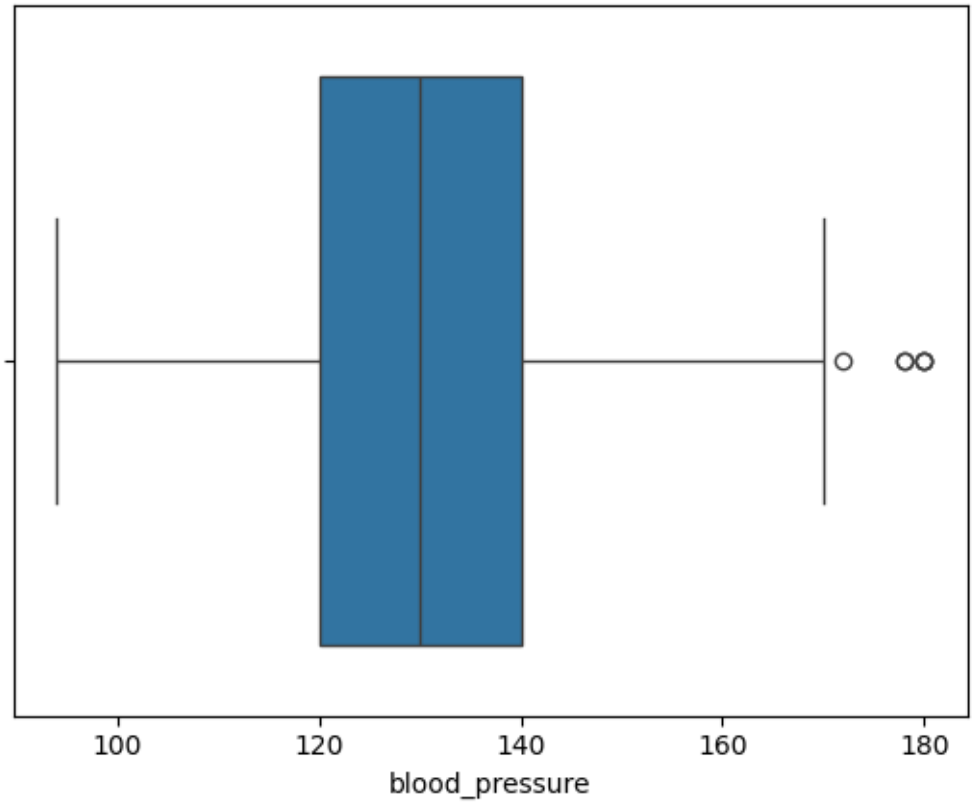
[ ]:
```
plt.figure(figsize=(15,20),facecolor='white')
plotnumber=1
```

```
for column in df.drop('Outcome',axis=1):
  if plotnumber<=14:
    ax=plt.subplot(3,5,plotnumber)
    sns.boxplot(df[column])
    plt.xlabel(column,fontsize=20)
    plt.ylabel('count',fontsize=20)
  plotnumber+=1
plt.tight_layout()
```



[ ]:
```
sns.boxplot(x='blood_pressure',data=df)
```

[ ]: <Axes: xlabel='blood_pressure'>



Handling the Outliers.

[ ]:
```
Q1=df['blood_pressure'].quantile(0.25)
print("low quartile:",Q1)
Q3=df['blood_pressure'].quantile(0.75)
print("low quartile:",Q3)
```

```
low quartile: 120.0
low quartile: 140.0
```

[ ]:
```
iqr=Q3-Q1
iqr
```

[ ]: 20.0

[ ]:
```
lower_limit=Q1-1.5*iqr
print("lower limit",lower_limit)
upper_limit=Q3+1.5*iqr
print("upper_limit",upper_limit)
```

```
lower limit 90.0
upper_limit 170.0
```

[ ]:
```
df.loc[df['blood_pressure']<lower_limit]
```

[ ]:
```
Empty DataFrame
Columns: [Outcome, st_segment, thal, blood_pressure, chest_pain_type,
num_major_vessels, sugar, ekg_result, cholestrol, st_depression, sex, age,
max_heartrate, exercise]
Index: []
```

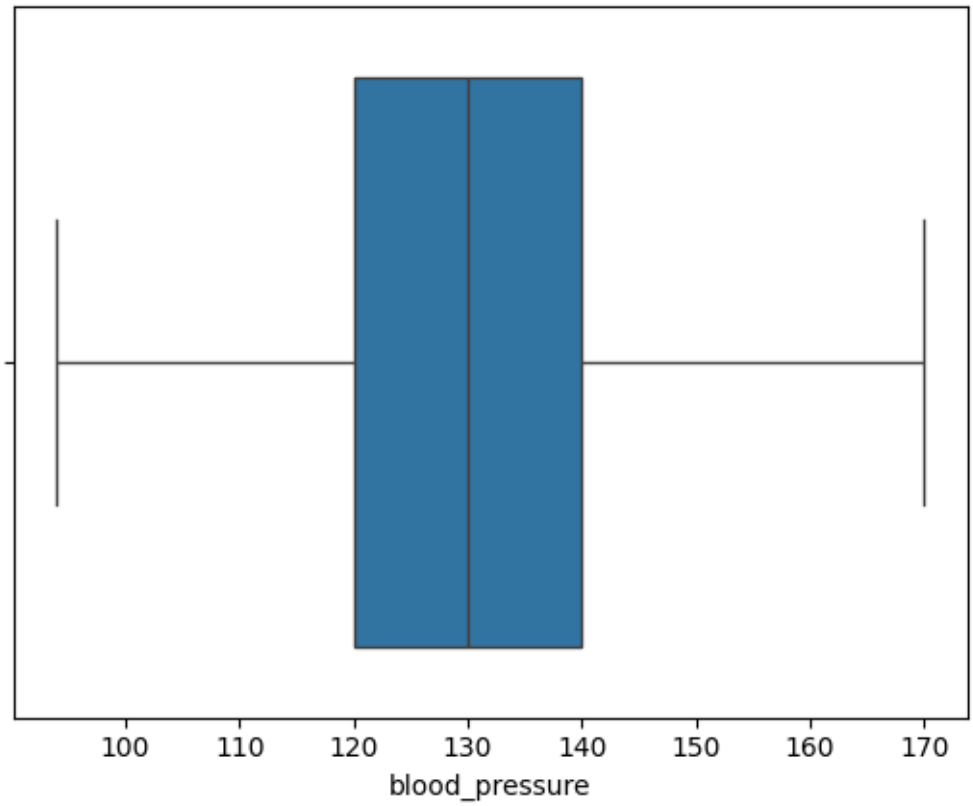
[ ]:
```
df.loc[df['blood_pressure']>upper_limit]
```

[ ]:
```
     Outcome  st_segment  thal  blood_pressure  chest_pain_type  \
4          0           3     2             178                1
33         0           1     1             180                4
72         0           1     2             172                3
75         1           2     2             178                4
113        1           2     2             180                3
176        1           2     1             180                4

     num_major_vessels  sugar  ekg_result  cholestrol  st_depression  sex  \
4                    0      0           2         270            4.2    1
33                   0      0           0         325            0.0    0
72                   0      1           0         199            0.5    1
75                   2      1           0         228            1.0    0
113                  0      1           2         274            1.6    1
176                  0      0           1         327            3.4    0

     age  max_heartrate  exercise
4     59            145         0
33    64            154         1
72    52            162         0
75    66            165         1
113   68            150         1
176   55            117         1
```

[ ]:
```
df.loc[df['blood_pressure']>upper_limit,'blood_pressure']=df['blood_pressure'].
↪median()
```

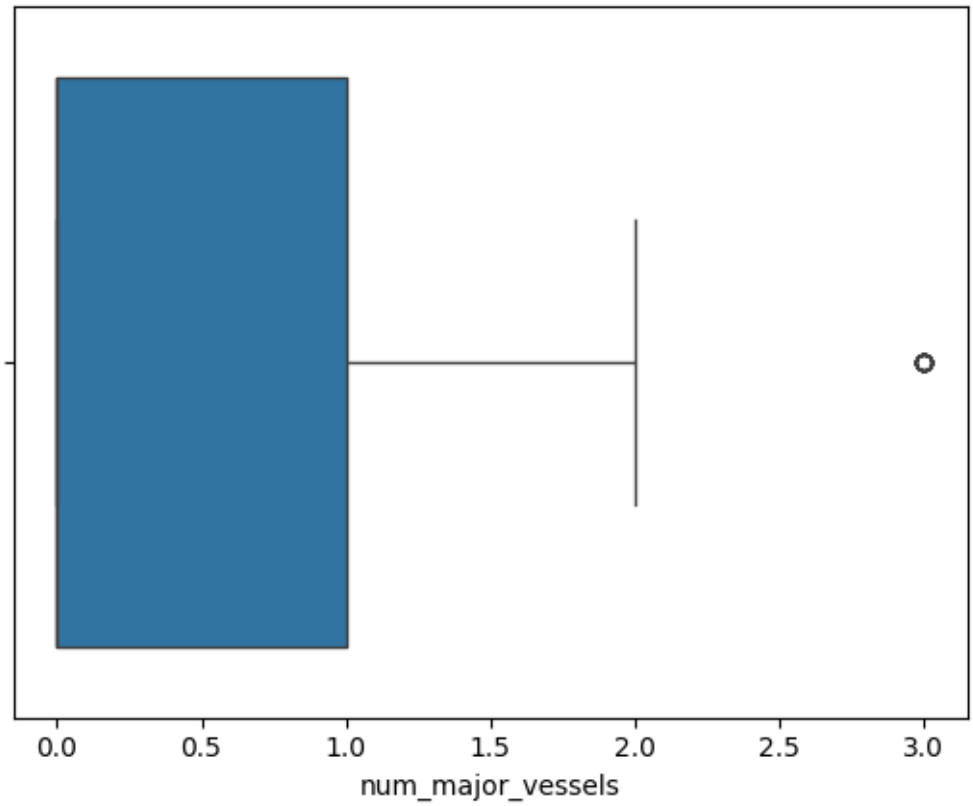
[ ]:
```
sns.boxplot(x='blood_pressure',data=df)
```

[ ]:
```
<Axes: xlabel='blood_pressure'>
```



[ ]:
```
sns.boxplot(x='num_major_vessels',data=df)
```

[ ]:
```
<Axes: xlabel='num_major_vessels'>
```



Handling the Outliers.

[ ]:
```
Q1=df['num_major_vessels'].quantile(0.25)
print("low quartile:",Q1)
Q3=df['num_major_vessels'].quantile(0.75)
print("low quartile:",Q3)
```

```
low quartile: 0.0
low quartile: 1.0
```

[ ]:
```
iqr=Q3-Q1
iqr
```

[ ]:
```
1.0
```

[ ]:
```
lower_limit=Q1-1.5*iqr
print("lower limit",lower_limit)
upper_limit=Q3+1.5*iqr
print("upper_limit",upper_limit)
```

```
lower limit -1.5
upper_limit 2.5
```

```
[ ]: df.loc[df['num_major_vessels']<lower_limit]
```

```
[ ]: Empty DataFrame
     Columns: [Outcome, st_segment, thal, blood_pressure, chest_pain_type,
     num_major_vessels, sugar, ekg_result, cholestrol, st_depression, sex, age,
     max_heartrate, exercise]
     Index: []
```

```
[ ]: df.loc[df['num_major_vessels']>upper_limit]
```

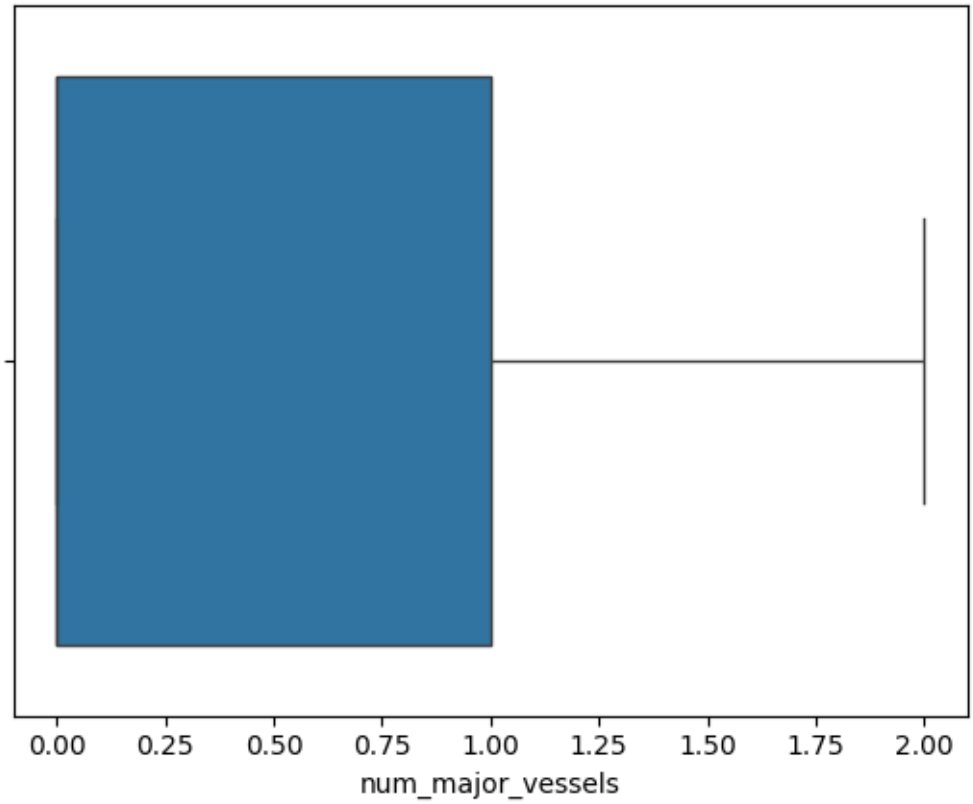
```
[ ]:      Outcome  st_segment  thal  blood_pressure  chest_pain_type  \
     2          1           1     1             125                4
     21         0           1     2             108                4
     31         1           2     2             150                4
     52         1           1     2             130                4
     66         1           2     1             130                4
     69         1           2     2             120                3
     85         1           1     1             118                3
     112        1           2     2             140                4
     119        1           2     1             138                4
     123        0           1     1             130                3
     124        1           2     1             160                4
     144        1           2     2             128                4
     162        1           3     2             160                4
     163        1           2     2             142                4

          num_major_vessels  sugar  ekg_result  cholestrol  st_depression  sex  \
     2                    3      0           2         304            0.0    1
     21                   3      1           0         233            0.1    1
     31                   3      0           2         225            1.0    0
     52                   3      1           2         330            1.8    1
     66                   3      0           2         322            2.4    1
     69                   3      0           0         188            2.0    1
     85                   3      0           2         149            0.8    1
     112                  3      0           0         298            4.2    1
     119                  3      1           0         294            1.9    0
     123                  3      1           2         246            0.0    1
     124                  3      0           2         286            1.5    1
     144                  3      0           2         216            2.2    1
     162                  3      0           2         164            6.2    0
     163                  3      0           2         309            0.0    1

          age  max_heartrate  exercise
     2     77            162         1
     21    52            147         0
     31    65            114         0
     52    63            132         1
```

66	70	109	0
69	49	139	0
85	49	126	0
112	51	122	1
119	62	106	0
123	53	173	0
124	67	108	1
144	58	131	1
162	62	145	0
163	45	147	1

[ ]:
```
df.
↪loc[df['num_major_vessels']>upper_limit,'num_major_vessels']=df['num_major_vessels'].
↪median()
```

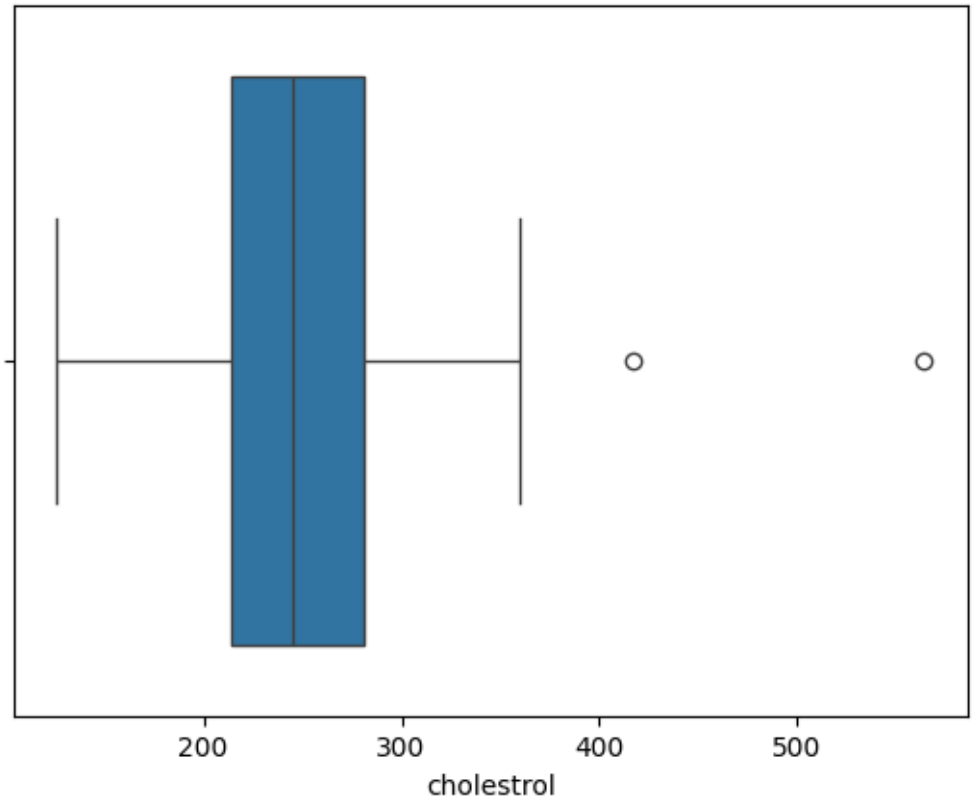
[ ]:
```
sns.boxplot(x='num_major_vessels',data=df)
```

[ ]:
```
<Axes: xlabel='num_major_vessels'>
```



[ ]:
```
sns.boxplot(x='cholestrol',data=df)
```

[ ]: <Axes: xlabel='cholestrol'>



Handling the Outliers.

[ ]:
```
Q1=df['cholestrol'].quantile(0.25)
print("low quartile:",Q1)
Q3=df['cholestrol'].quantile(0.75)
print("low quartile:",Q3)
```

```
low quartile: 213.75
low quartile: 281.25
```

[ ]:
```
iqr=Q3-Q1
iqr
```

[ ]: 67.5

[ ]:
```
lower_limit=Q1-1.5*iqr
print("lower limit",lower_limit)
upper_limit=Q3+1.5*iqr
print("upper_limit",upper_limit)
```

```
lower limit 112.5
upper_limit 382.5
```

[ ]:
```
df.loc[df['cholestrol']<lower_limit]
```

[ ]:
```
Empty DataFrame
Columns: [Outcome, st_segment, thal, blood_pressure, chest_pain_type,
num_major_vessels, sugar, ekg_result, cholestrol, st_depression, sex, age,
max_heartrate, exercise]
Index: []
```

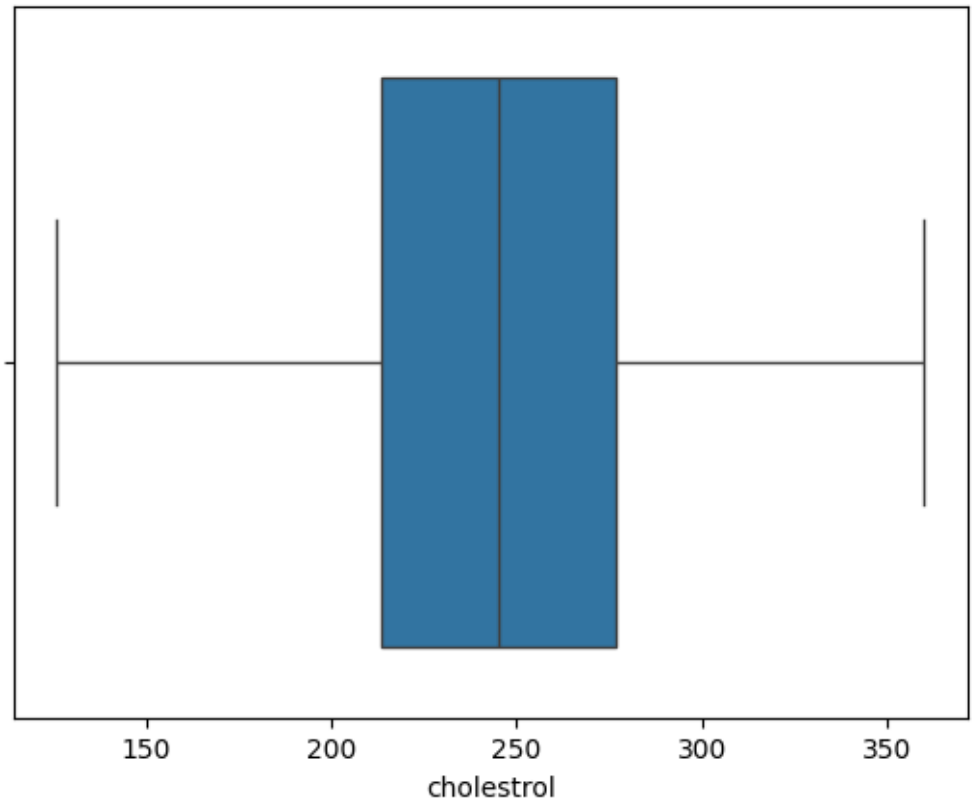
[ ]:
```
df.loc[df['cholestrol']>upper_limit]
```

[ ]:
```
    Outcome  st_segment  thal  blood_pressure  chest_pain_type  \
43        0           1     1             140                3
60        0           2     2             115                3

    num_major_vessels  sugar  ekg_result  cholestrol  st_depression  sex  age  \
43                  1      1           2         417            0.8    0   65
60                  0      0           2         564            1.6    0   67

    max_heartrate  exercise
43            157         0
60            160         0
```

[ ]:
```
df.loc[df['cholestrol']>upper_limit,'cholestrol']=df['cholestrol'].median()
```

[ ]:
```
sns.boxplot(x='cholestrol',data=df)
```

[ ]:
```
<Axes: xlabel='cholestrol'>
```



Feature Engineering

A correlation matrixis simply a table which displays the correlation coefficients for different variables. The matrix depicts the correlation between all the possible pairs of values in a table.

[ ]:
```
df.corr()
```

[ ]:

	Outcome	st_segment	thal	blood_pressure	\
Outcome	1.000000	0.344224	0.460933	0.076048	
st_segment	0.344224	1.000000	0.317019	0.061536	
thal	0.460933	0.317019	1.000000	0.122475	
blood_pressure	0.076048	0.061536	0.122475	1.000000	
chest_pain_type	0.412829	0.121207	0.254939	-0.037038	
num_major_vessels	0.300959	0.017118	0.168025	0.021469	
sugar	0.003379	0.050199	0.064897	0.096842	
ekg_result	0.145933	0.172191	-0.004791	0.113544	
cholestrol	0.146419	-0.047307	-0.074096	0.163281	
st_depression	0.382930	0.615948	0.304672	0.176657	
sex	0.335421	0.093340	0.412284	-0.016618	
age	0.138255	0.169918	0.067663	0.259479	
max_heartrate	-0.375352	-0.418102	-0.278681	-0.014901	
exercise	0.448647	0.225459	0.317990	0.056117	

	chest_pain_type	num_major_vessels	sugar	ekg_result
Outcome	0.412829	0.300959	0.003379	0.145933
st_segment	0.121207	0.017118	0.050199	0.172191
thal	0.254939	0.168025	0.064897	-0.004791
blood_pressure	-0.037038	0.021469	0.096842	0.113544
chest_pain_type	1.000000	0.117806	-0.088992	0.033379
num_major_vessels	0.117806	1.000000	0.119789	0.006678
sugar	-0.088992	0.119789	1.000000	0.053864
ekg_result	0.033379	0.006678	0.053864	1.000000
cholestrol	0.081077	0.125992	0.001109	0.141028
st_depression	0.080799	0.082364	-0.039055	0.097321
sex	0.086057	0.030801	0.066010	0.045786
age	0.085001	0.332616	0.176101	0.126856
max_heartrate	-0.301792	-0.128077	0.058369	-0.102766
exercise	0.346266	0.129809	-0.005956	0.037773

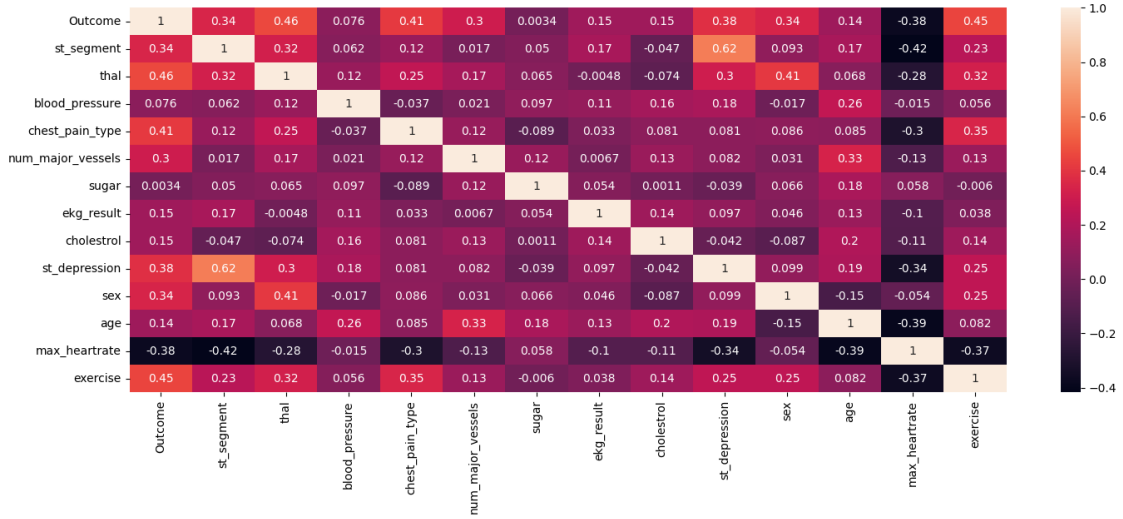
	cholestrol	st_depression	sex	age
Outcome	0.146419	0.382930	0.335421	0.138255
st_segment	-0.047307	0.615948	0.093340	0.169918
thal	-0.074096	0.304672	0.412284	0.067663
blood_pressure	0.163281	0.176657	-0.016618	0.259479
chest_pain_type	0.081077	0.080799	0.086057	0.085001
num_major_vessels	0.125992	0.082364	0.030801	0.332616
sugar	0.001109	-0.039055	0.066010	0.176101
ekg_result	0.141028	0.097321	0.045786	0.126856
cholestrol	1.000000	-0.042138	-0.087374	0.200082
st_depression	-0.042138	1.000000	0.099374	0.189700
sex	-0.087374	0.099374	1.000000	-0.148997
age	0.200082	0.189700	-0.148997	1.000000
max_heartrate	-0.108363	-0.341045	-0.053960	-0.394630
exercise	0.137476	0.249167	0.251096	0.081811

	max_heartrate	exercise
Outcome	-0.375352	0.448647
st_segment	-0.418102	0.225459
thal	-0.278681	0.317990
blood_pressure	-0.014901	0.056117
chest_pain_type	-0.301792	0.346266
num_major_vessels	-0.128077	0.129809
sugar	0.058369	-0.005956
ekg_result	-0.102766	0.037773
cholestrol	-0.108363	0.137476
st_depression	-0.341045	0.249167
sex	-0.053960	0.251096
age	-0.394630	0.081811
max_heartrate	1.000000	-0.365065

```
exercise                -0.365065   1.000000
```

[ ]:
```
plt.figure(figsize=(17,6))
sns.heatmap(df.corr(),annot=True)
```

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



Model Creation

Training a machine learning model for the task of heart disease prediction.

Using Logistic Regression: Logistic regression is a supervised machine learning algorithm used for classification tasks. Its primary goal is to predict the probability that an instance belongs to a given class or not.

[ ]:
```
x=df.drop(columns='Outcome',axis=1)
y=df['Outcome']
```

[ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
  ↪20,random_state=42)
```

[ ]:
```
print(x.shape,x_train.shape,x_test.shape)
```

```
(180, 13) (144, 13) (36, 13)
```

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

[ ]:
```
LogisticRegression()
```

[ ]:
```
x_train_prediction=model.predict(x_train)
training_data_accuracy=accuracy_score(x_train_prediction,y_train)
```

[ ]:
```
print("accuracy on training data:",training_data_accuracy)
```

```
accuracy on training data: 0.8541666666666666
```

[ ]:
```
test_predictions = model.predict(x_test)
test_accuracy = accuracy_score(y_test, test_predictions)
print("Testing Accuracy:", test_accuracy)
```

```
Testing Accuracy: 0.8333333333333334
```

Smote Logistic Regression:

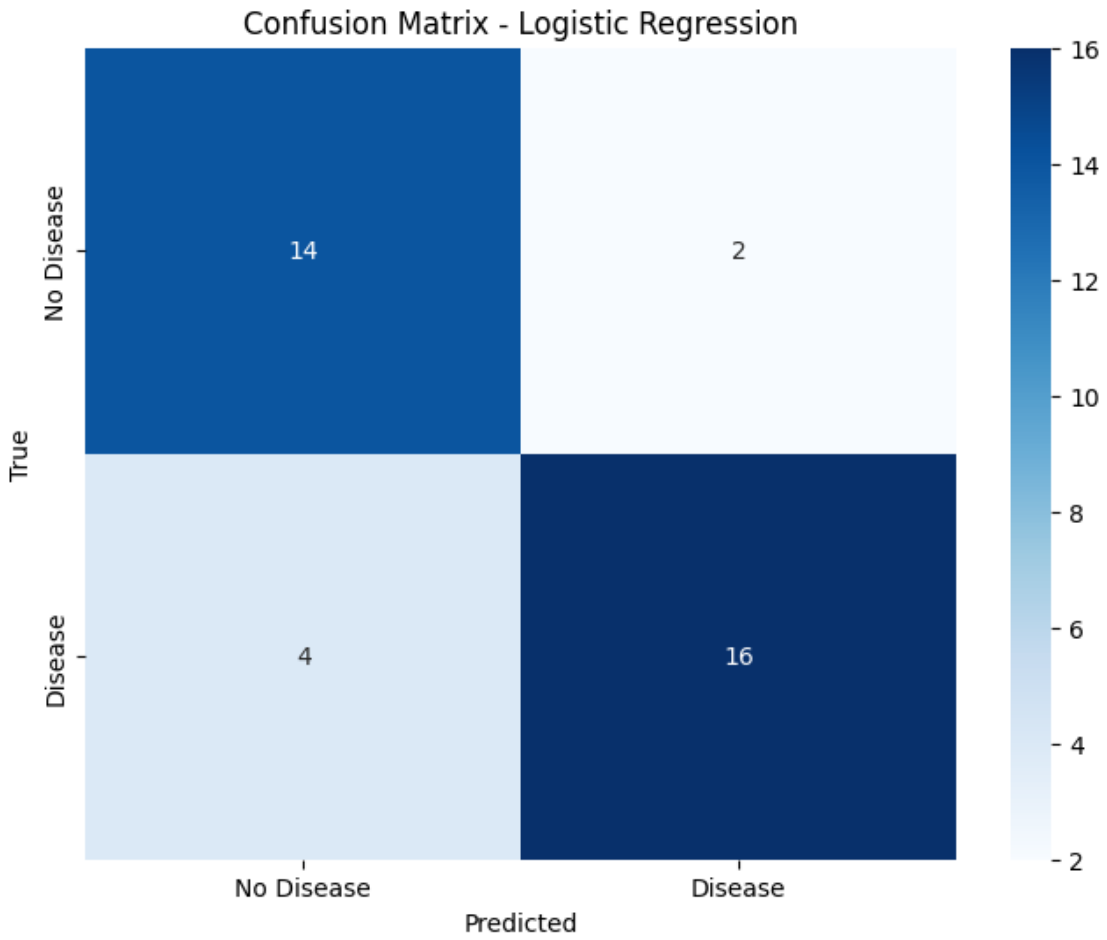
[ ]:
```
from imblearn.over_sampling import SMOTE
sm=SMOTE()
x_smote,y_smote=sm.fit_resample(x_train,y_train)
y_smote.value_counts()
```

[ ]:
```
1    84
0    84
Name: Outcome, dtype: int64
```

[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
lr= LogisticRegression()
lr.fit(x_smote, y_smote)
y_pred_smote=lr.predict(x_test)
accuracy_score(y_test,y_pred_smote)
print("Smote Accuracy : {:.2f}%".
 ↪format(accuracy_score(y_test,y_pred_smote)*100))
```

```
Smote Accuracy : 80.56%
```

[ ]:
```
from sklearn.metrics import confusion_matrix
clf = LogisticRegression()
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=['No Disease',␣
 ↪'Disease'], yticklabels=['No Disease', 'Disease'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()
```



[ ]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8333333333333334
```

[ ]:
```
from sklearn.metrics import precision_score, recall_score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
print("Precision:", precision)
print("Recall:", recall)
```

```
Precision: 0.8888888888888888
Recall: 0.8
```

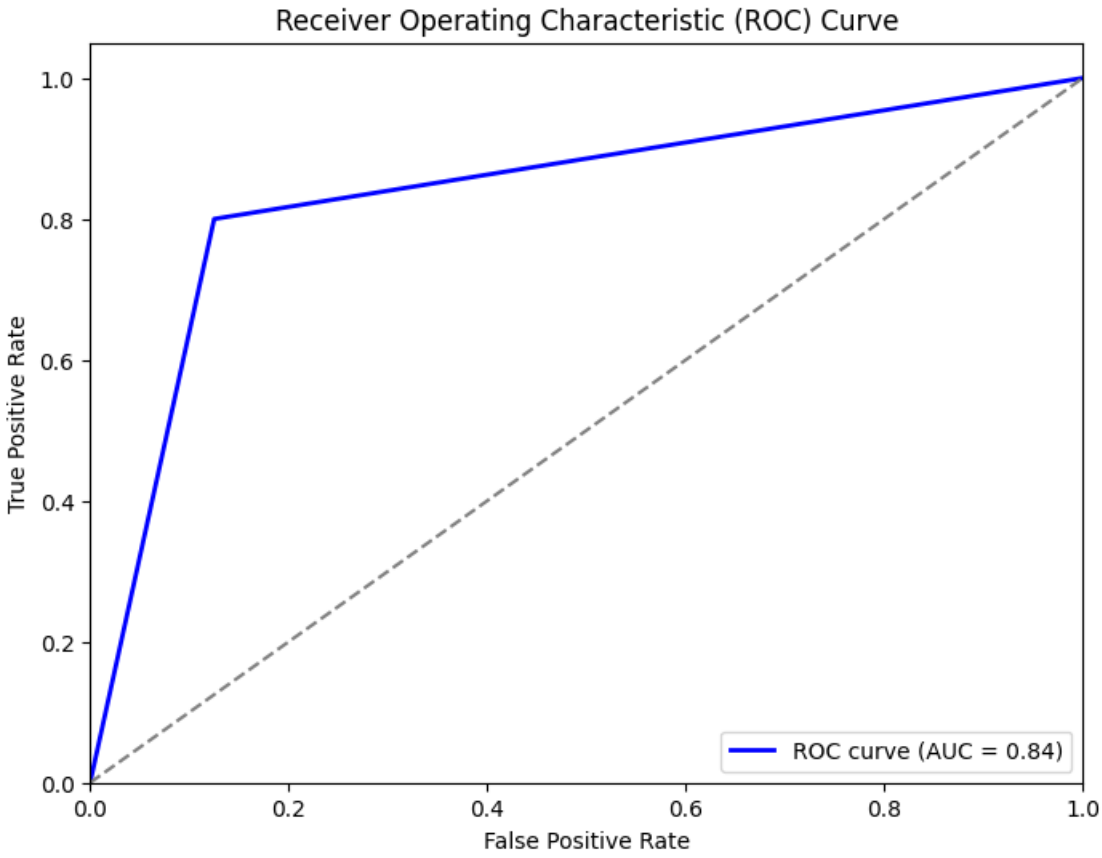
[ ]:
```
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("F1 Score:", f1)
```

```
F1 Score: 0.8421052631578948
```

[ ]:
```
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.88      0.82        16
           1       0.89      0.80      0.84        20

    accuracy                           0.83        36
   macro avg       0.83      0.84      0.83        36
weighted avg       0.84      0.83      0.83        36
```

[ ]:
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)' %␣
 ↪roc_auc)
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



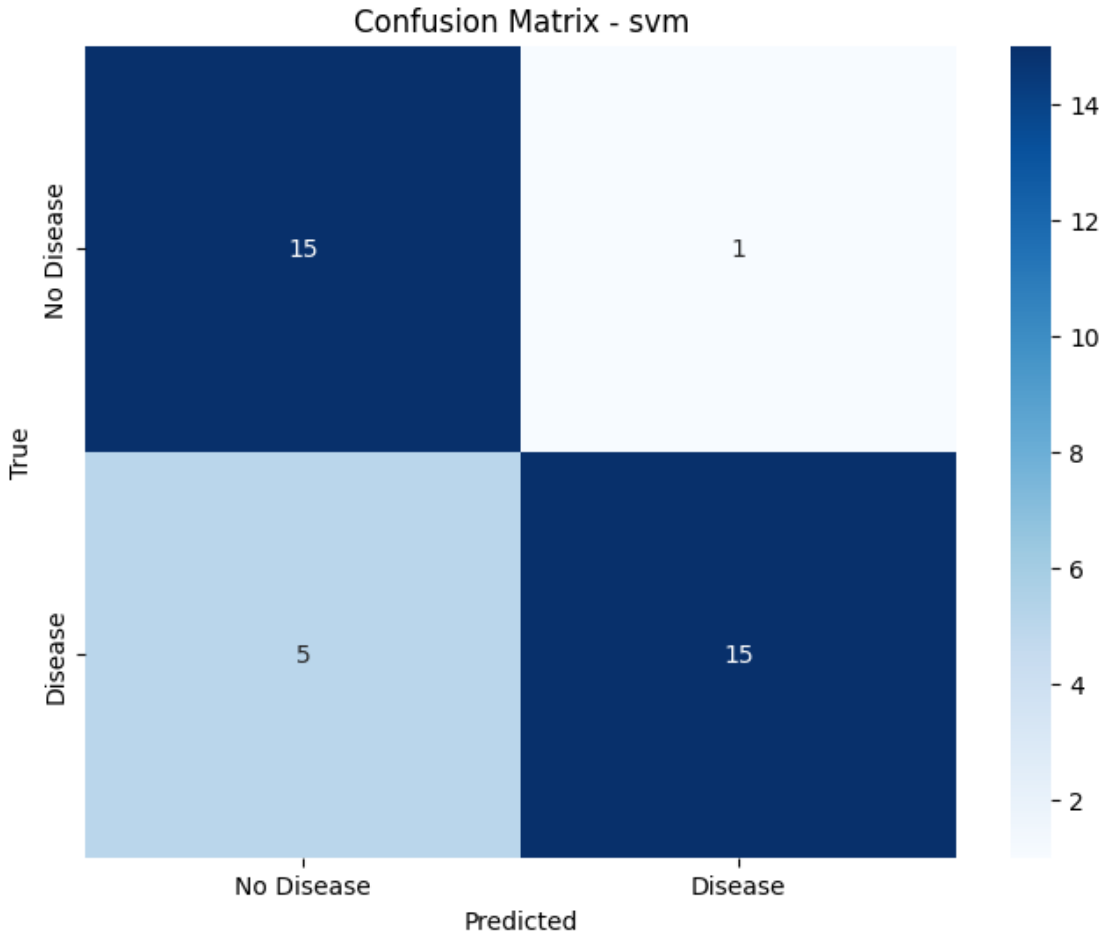
Support Vector Machine(SVM): Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks. SVM tries to finds the "best" margin (distance between the line and the support vectors) that separates the classes and this reduces the risk of error on the data when compared to Logistic regression.

[ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
X = df.drop('Outcome', axis=1)
y = df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = SVC(kernel='linear', random_state=42)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

```
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8611111111111112
```

[ ]:
```
from sklearn.metrics import confusion_matrix
clf = LogisticRegression()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=['No Disease',␣
↪'Disease'], yticklabels=['No Disease', 'Disease'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - svm')
plt.show()
```



[ ]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8333333333333334
```

[ ]:
```
from sklearn.metrics import precision_score, recall_score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
print("Precision:", precision)
print("Recall:", recall)
```

```
Precision: 0.9375
Recall: 0.75
```

[ ]:
```
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("F1 Score:", f1)
```

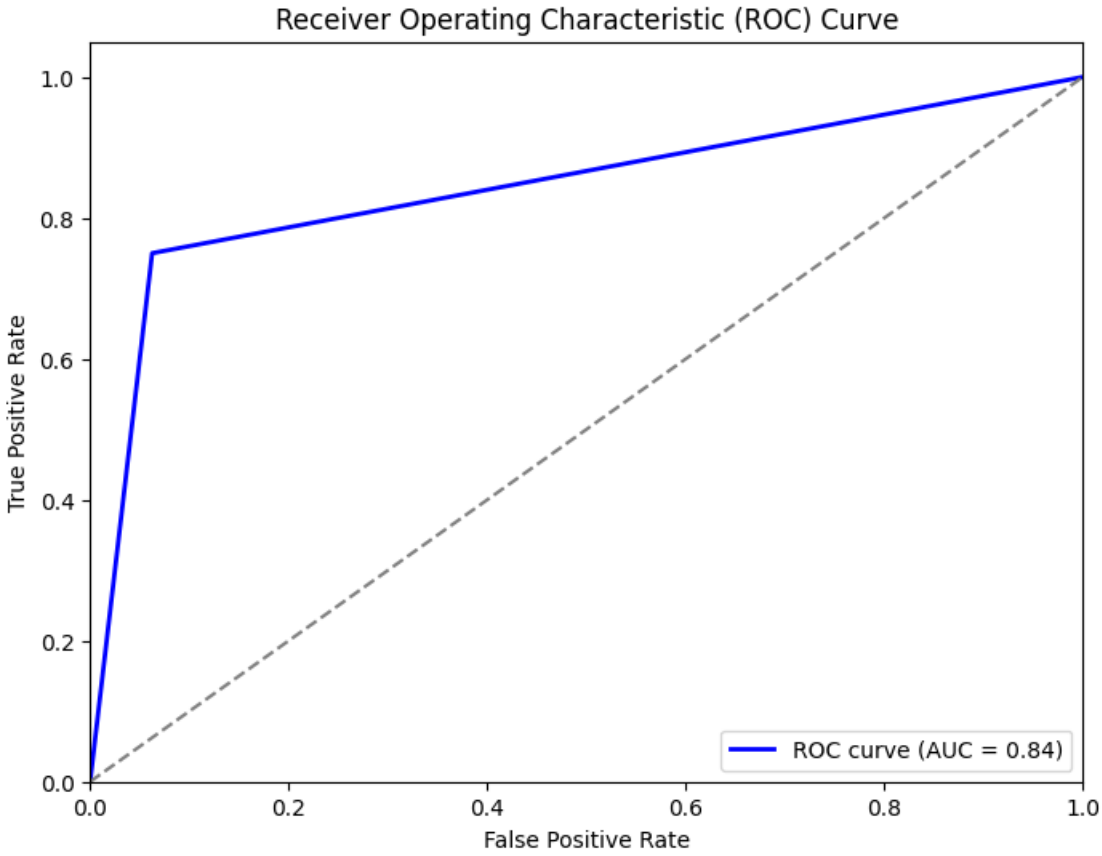
```
F1 Score: 0.8333333333333334
```

[ ]:
```
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.94      0.83        16
           1       0.94      0.75      0.83        20

    accuracy                           0.83        36
   macro avg       0.84      0.84      0.83        36
weighted avg       0.85      0.83      0.83        36
```

[ ]:
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)' %␣
 ↪roc_auc)
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



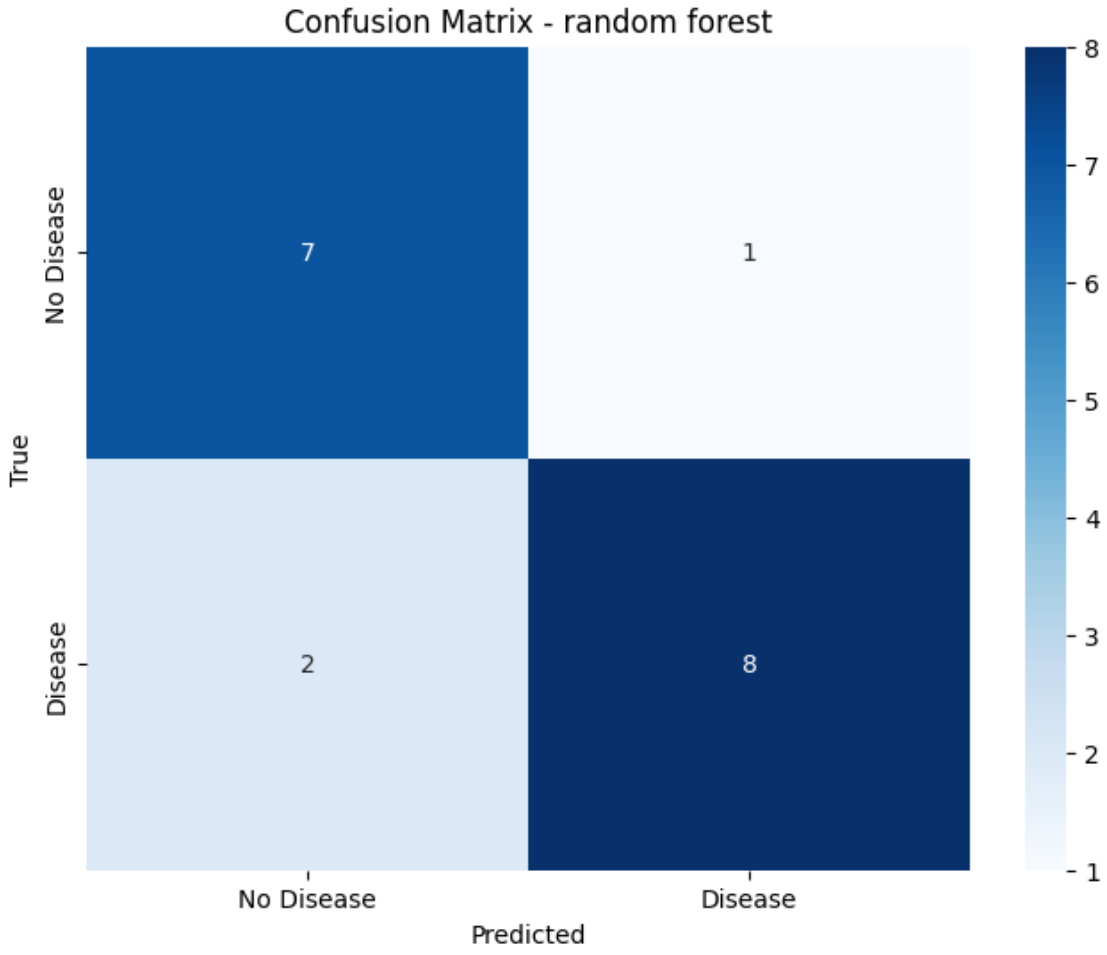
Random Forest: Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
data = df.replace('?', pd.NA).dropna()
X = df.drop('Outcome', axis=1)
y = df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,␣
 ↪random_state=42)
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8888888888888888
```

[ ]:
```
from sklearn.metrics import confusion_matrix
clf = LogisticRegression()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=['No Disease',␣
↪'Disease'], yticklabels=['No Disease', 'Disease'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - random forest')
plt.show()
```



[ ]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8333333333333334
```

[ ]:
```
from sklearn.metrics import precision_score, recall_score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
print("Precision:", precision)
print("Recall:", recall)
```

```
Precision: 0.8888888888888888
Recall: 0.8
```

[ ]:
```
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("F1 Score:", f1)
```

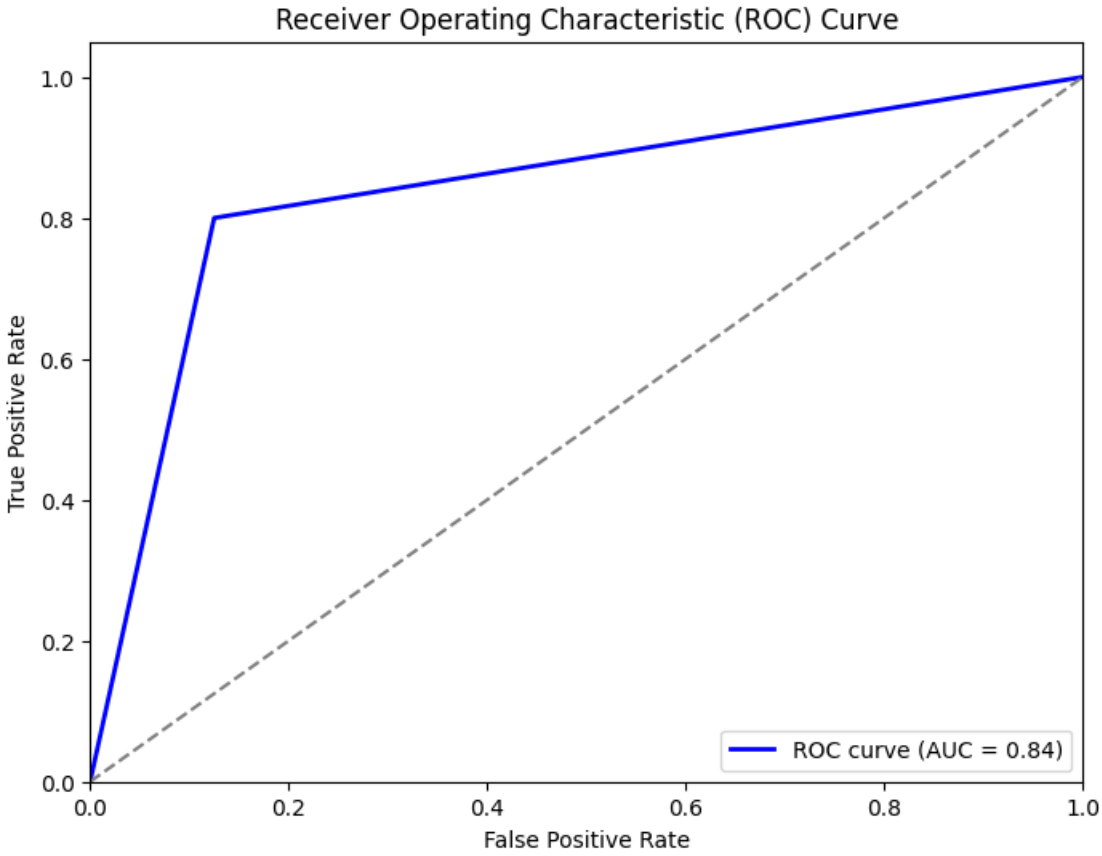
```
F1 Score: 0.8421052631578948
```

[ ]:
```
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.88      0.82         8
           1       0.89      0.80      0.84        10

    accuracy                           0.83        18
   macro avg       0.83      0.84      0.83        18
weighted avg       0.84      0.83      0.83        18
```

[ ]:
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)' %␣
 ↪roc_auc)
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



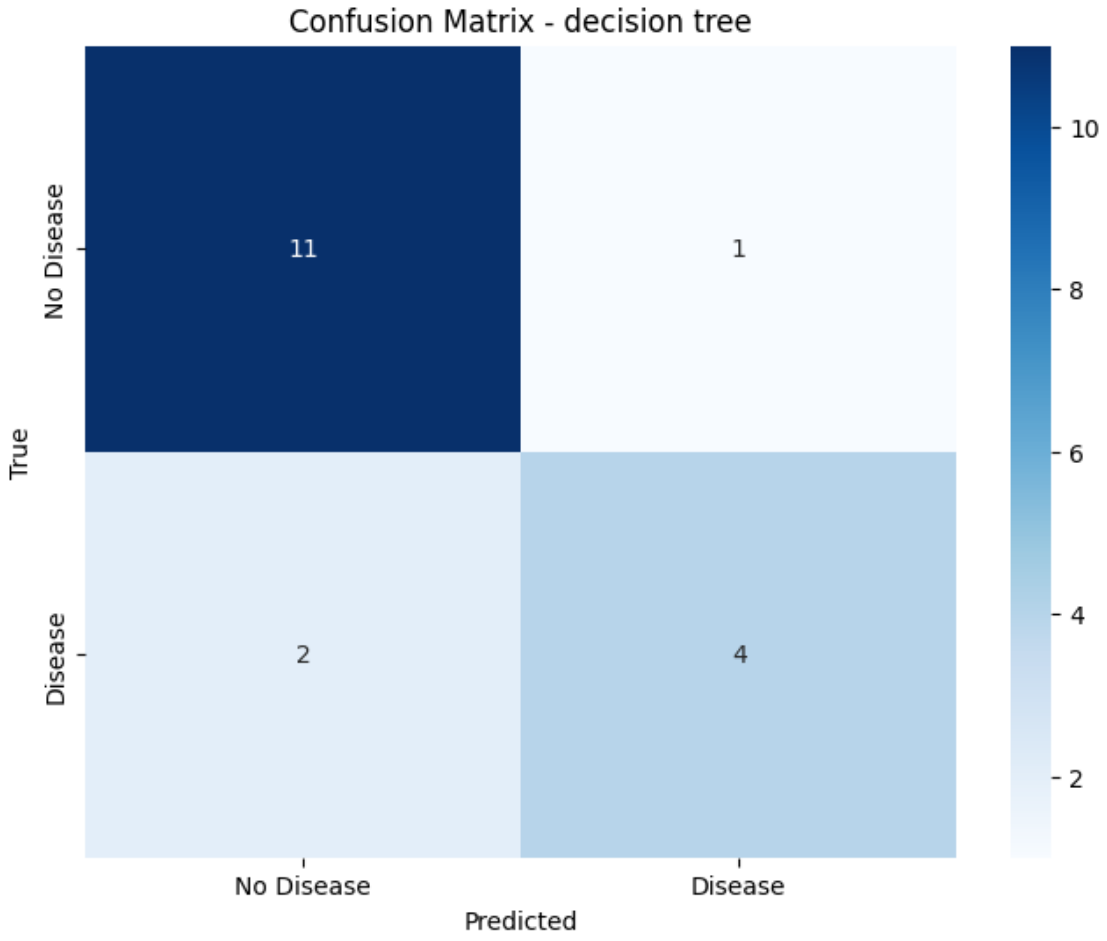
decision tree:

Decision trees are a type of machine-learning algorithm that can be used for both classification and regression tasks. They work by learning simple decision rules inferred from the data features. These rules can then be used to predict the value of the target variable for new data samples.

[ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,␣
 ↪random_state=4)
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7777777777777778
```

[ ]:
```
from sklearn.metrics import confusion_matrix
clf = LogisticRegression()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=['No Disease',␣
↪'Disease'], yticklabels=['No Disease', 'Disease'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - decision tree')
plt.show()
```



[ ]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8333333333333334
```

[ ]:
```
from sklearn.metrics import precision_score, recall_score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
print("Precision:", precision)
print("Recall:", recall)
```

```
Precision: 0.8
Recall: 0.6666666666666666
```

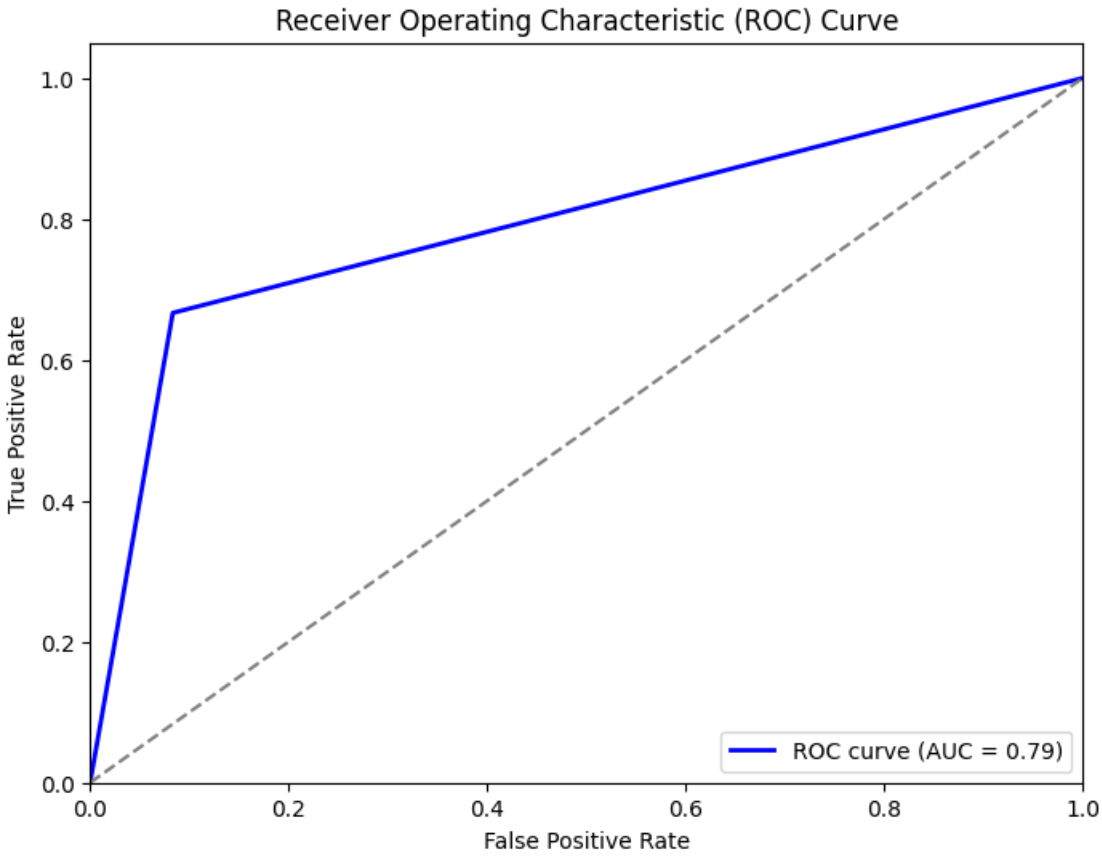
[ ]:
```
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("F1 Score:", f1)
```

```
F1 Score: 0.7272727272727272
```

[ ]:
```
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.92      0.88        12
           1       0.80      0.67      0.73         6

    accuracy                           0.83        18
   macro avg       0.82      0.79      0.80        18
weighted avg       0.83      0.83      0.83        18
```

[ ]:
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)' %␣
 ↪roc_auc)
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



K Nearest Neighbors(KNN):

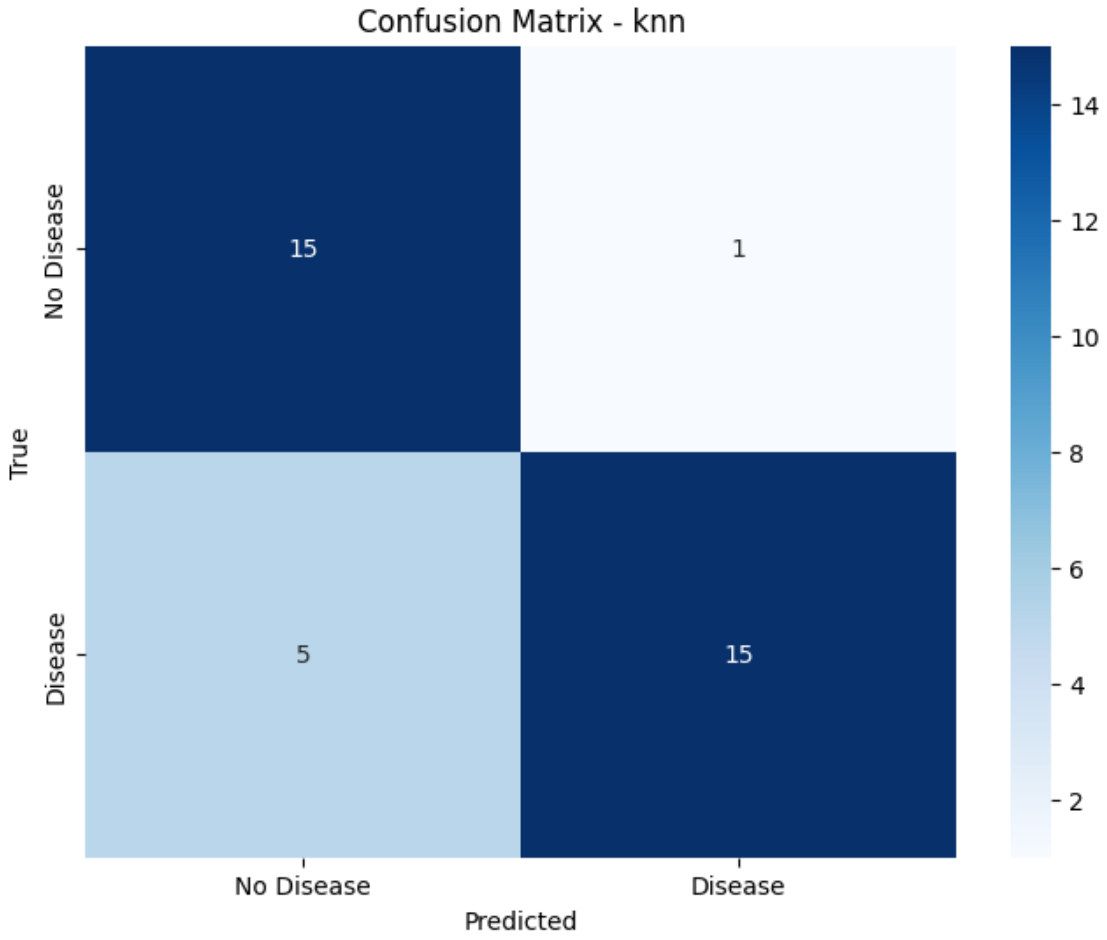
The abbreviation KNN stands for "K-Nearest Neighbour". It is a supervised machine learning algorithm. The algorithm can be used to solve both classification and regression problem statements. The number of nearest neighbours to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'.

[ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
X = df.drop('Outcome', axis=1)
y = df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8611111111111112
```

[ ]:
```
from sklearn.metrics import confusion_matrix
clf = LogisticRegression()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=['No Disease',␣
↪'Disease'], yticklabels=['No Disease', 'Disease'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - knn')
plt.show()
```



[ ]:
```
from sklearn.metrics import accuracy_score
# Assuming y_test and y_pred are your true labels and predicted labels␣
 ↪respectively
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8333333333333334
```

[ ]:
```
from sklearn.metrics import precision_score, recall_score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
print("Precision:", precision)
print("Recall:", recall)
```

```
Precision: 0.9375
Recall: 0.75
```

[ ]:
```
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("F1 Score:", f1)
```

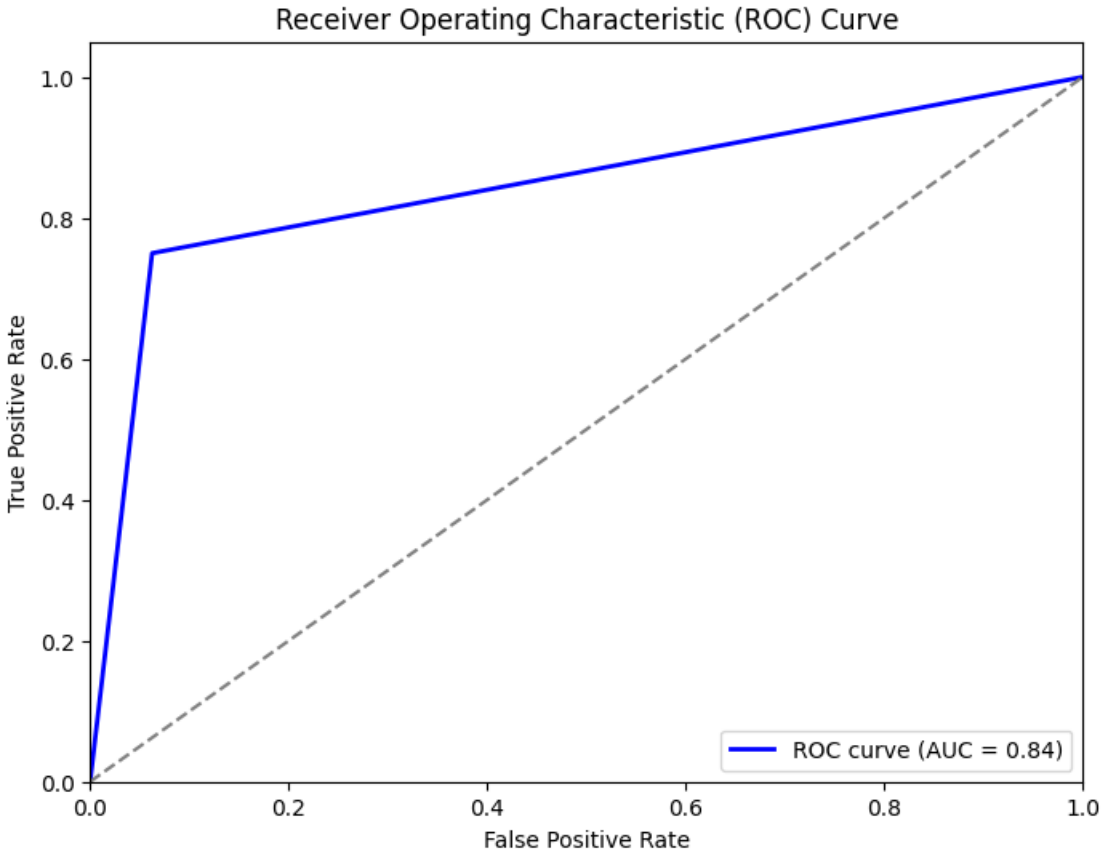
```
F1 Score: 0.8333333333333334
```

[ ]:
```
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.94      0.83        16
           1       0.94      0.75      0.83        20

    accuracy                           0.83        36
   macro avg       0.84      0.84      0.83        36
weighted avg       0.85      0.83      0.83        36
```

[ ]:
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)' %␣
 ↪roc_auc)
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



Conclusion:

This project predicts people with cardiovascular disease by extracting the patient medical history that leads to a fatal heart disease from a dataset that includes patients' medical history such as chest pain, sugar level, blood pressure, etc.Finally, we can conclude that real-time predictors will be essential in the healthcare sector nowadays. From this project, we will be able to predict heart disease using the patient's data from the model using the Logistic Regression, Support vector machine, Decision Tree Algorithm, Random forest, KNN, thereby making accurate heart disease prediction using machine learning.

The Random Forest algorithm achieved the highest accuracy in predicting heart disease, with a rate of 88.9%, followed by the Support Vector Machine and K-nearest neighbor algorithm at 86.11%