Heart Disease (including Coronary Heart Disease, Hypertension, and Stroke) remains the No. 1 cause of death in the US.The Heart Disease and Stroke Statistics—2019 Update from the American Heart Association indicates that:

In this machine learning project, we have collected the dataset from UCI (https://archive.ics.uci.edu/ml/datasets/statlog+(heart)) and we will be using Machine Learning to make predictions on whether a person is suffering from Heart Disease or not.

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

%matplotlib inline


import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage.
It includes your credentials.
# You might want to remove those credentials before you share the
notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='5i6ow0P4tpnCXrwPmHgjOlvS_22JMaJwUe39DOIQ-wz8',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-
storage.appdomain.cloud')

bucket = 'visualizingandpredictingheartdise-donotdelete-pr-
dhoozkeqjwqosc'
object_key = 'Heart_Disease_Prediction.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like
object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType(
__iter__, body )

data = pd.read_csv(body)
data.head()
```

```
   Age  Sex  Chest pain type   BP  Cholesterol  FBS over 120  EKG
results  \
0   70    1                4  130          322             0
2
1   67    0                3  115          564             0
2
2   57    1                2  124          261             0
0
3   64    1                4  128          263             0
0
4   74    0                2  120          269             0
2

   Max HR  Exercise angina  ST depression  Slope of ST  \
0     109                 0            2.4            2
1     160                 0            1.6            2
2     141                 0            0.3            1
3     105                 1            0.2            2
4     121                 1            0.2            1

   Number of vessels fluro  Thallium Heart Disease
0                        3         3      Presence
1                        0         7       Absence
2                        0         7      Presence
3                        1         7       Absence
4                        1         3       Absence

data.tail()

     Age  Sex  Chest pain type   BP  Cholesterol  FBS over 120  EKG
results  \
265   52    1                3  172          199             1
0
266   44    1                2  120          263             0
0
267   56    0                2  140          294             0
2
268   57    1                4  140          192             0
0
269   67    1                4  160          286             0
2

     Max HR  Exercise angina  ST depression  Slope of ST  \
265     162                 0            0.5            1
266     173                 0            0.0            1
267     153                 0            1.3            2
268     148                 0            0.4            2
269     108                 1            1.5            2

     Number of vessels fluro  Thallium Heart Disease
```

```
265                        0         7       Absence
266                        0         7       Absence
267                        0         3       Absence
268                        0         6       Absence
269                        3         3      Presence
```

```
print(f'No of Rows in the dataset : {data.shape[0]}')
print(f'No of Columns in the dataset : {data.shape[1]}')
```

```
No of Rows in the dataset : 270
No of Columns in the dataset : 14
```

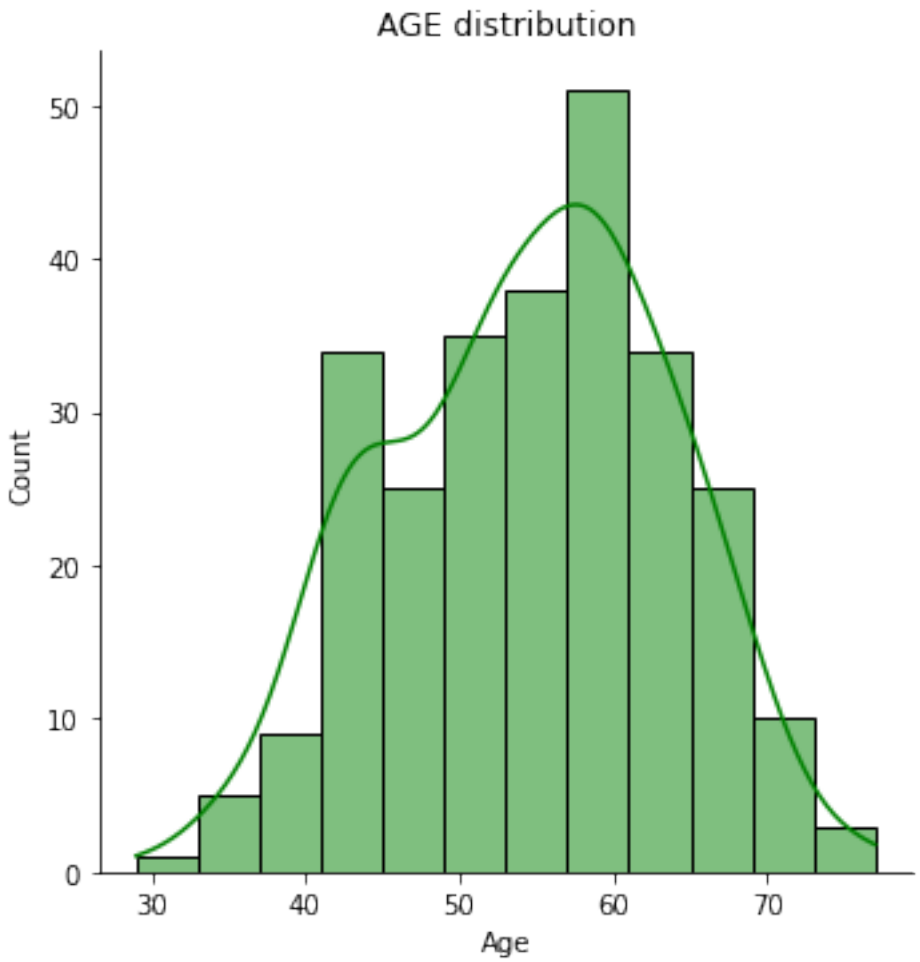
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Age                       270 non-null    int64  
 1   Sex                       270 non-null    int64  
 2   Chest pain type           270 non-null    int64  
 3   BP                        270 non-null    int64  
 4   Cholesterol               270 non-null    int64  
 5   FBS over 120              270 non-null    int64  
 6   EKG results               270 non-null    int64  
 7   Max HR                    270 non-null    int64  
 8   Exercise angina           270 non-null    int64  
 9   ST depression             270 non-null    float64
 10  Slope of ST               270 non-null    int64  
 11  Number of vessels fluro   270 non-null    int64  
 12  Thallium                  270 non-null    int64  
 13  Heart Disease             270 non-null    object 
dtypes: float64(1), int64(12), object(1)
memory usage: 29.7+ KB
```

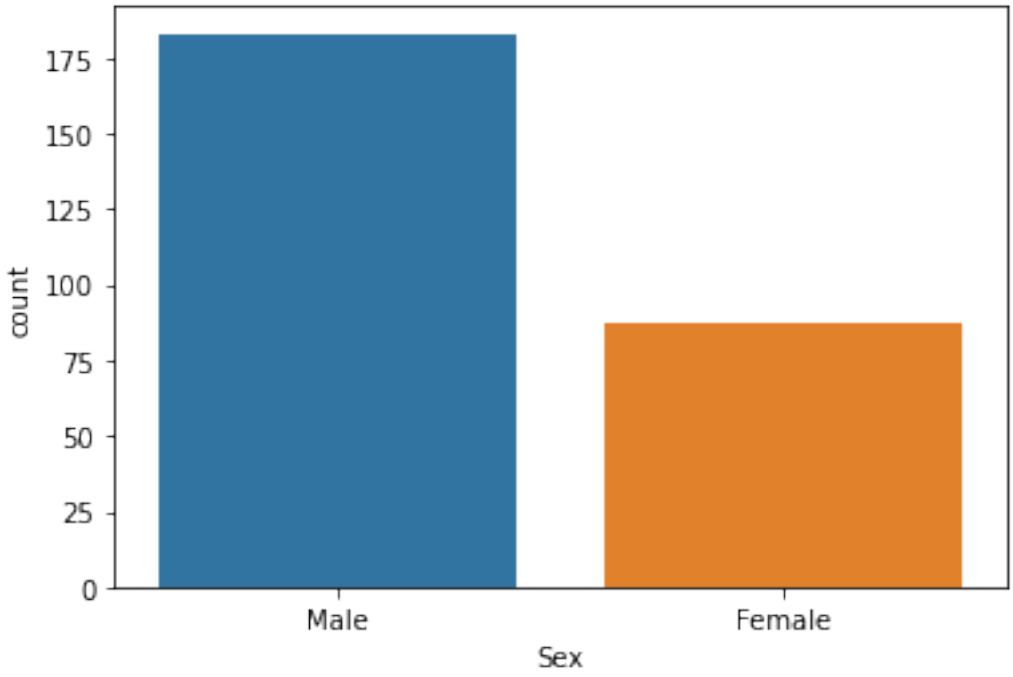
```
data.describe()
```

```
              Age         Sex  Chest pain type          BP
Cholesterol  \
count  270.000000  270.000000       270.000000  270.000000
270.000000
mean    54.433333    0.677778         3.174074  131.344444
249.659259
std      9.109067    0.468195         0.950090   17.861608
51.686237
min     29.000000    0.000000         1.000000   94.000000
126.000000
25%     48.000000    0.000000         3.000000  120.000000
213.000000
50%     55.000000    1.000000         3.000000  130.000000
```

```
245.000000
75%     61.000000    1.000000         4.000000  140.000000
280.000000
max     77.000000    1.000000         4.000000  200.000000
564.000000

       FBS over 120  EKG results      Max HR  Exercise angina  ST
depression  \
count    270.000000   270.000000  270.000000       270.000000
270.00000
mean       0.148148     1.022222  149.677778         0.329630
1.05000
std        0.355906     0.997891   23.165717         0.470952
1.14521
min        0.000000     0.000000   71.000000         0.000000
0.00000
25%        0.000000     0.000000  133.000000         0.000000
0.00000
50%        0.000000     2.000000  153.500000         0.000000
0.80000
75%        0.000000     2.000000  166.000000         1.000000
1.60000
max        1.000000     2.000000  202.000000         1.000000
6.20000

       Slope of ST  Number of vessels fluro    Thallium
count   270.000000               270.000000  270.000000
mean      1.585185                 0.670370    4.696296
std       0.614390                 0.943896    1.940659
min       1.000000                 0.000000    3.000000
25%       1.000000                 0.000000    3.000000
50%       2.000000                 0.000000    3.000000
75%       2.000000                 1.000000    7.000000
max       3.000000                 3.000000    7.000000

data.columns

Index(['Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol', 'FBS over
120',
       'EKG results', 'Max HR', 'Exercise angina', 'ST depression',
       'Slope of ST', 'Number of vessels fluro', 'Thallium', 'Heart
Disease'],
      dtype='object')

sns.displot(x=data["Age"], kde=True, color='green')
plt.title("AGE distribution");
```



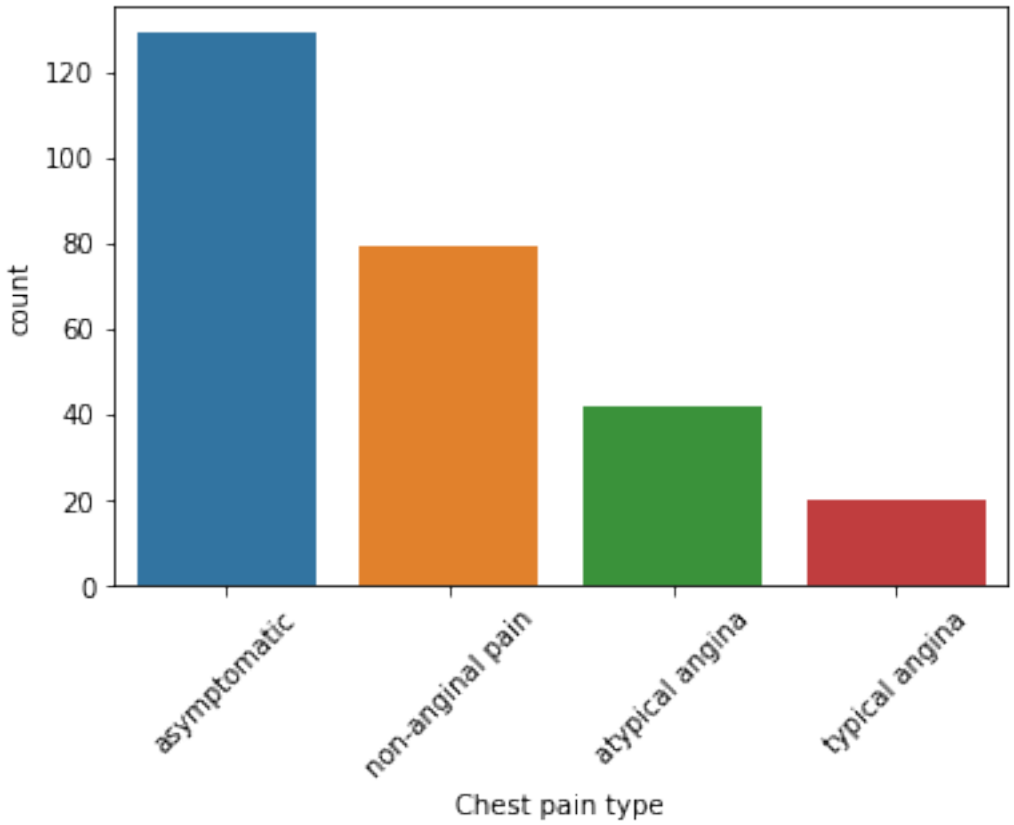

```
labels = ['Male', 'Female']
order = data['Sex'].value_counts().index

sns.countplot(x='Sex', data=data, order=order)
plt.xticks([0, 1], labels)
plt.show()
```



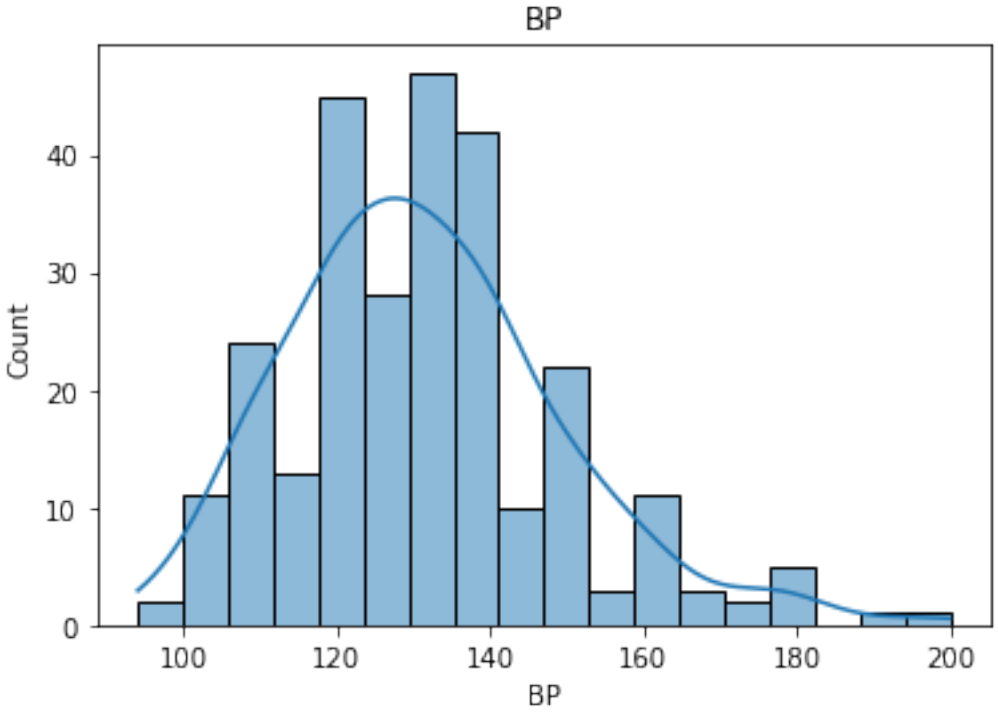

```
data['Chest pain type'].value_counts()
```

```
4    129
3     79
2     42
1     20
Name: Chest pain type, dtype: int64
```

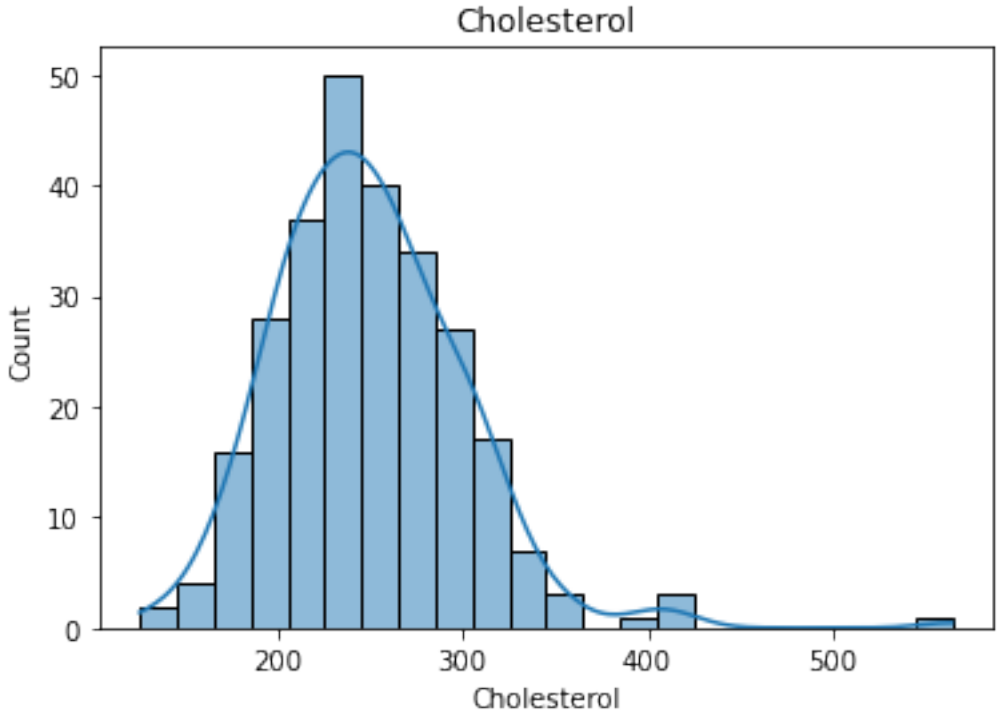
```
labels = ["asymptomatic","non-anginal pain","atypical angina","typical
angina"]
order = data['Chest pain type'].value_counts().index

sns.countplot(x='Chest pain type', data=data, order=order)
plt.xticks([0,1,2,3], labels, rotation=45)
plt.show()
```




```
sns.histplot(x=data["BP"],kde=True)
plt.title("BP");
```



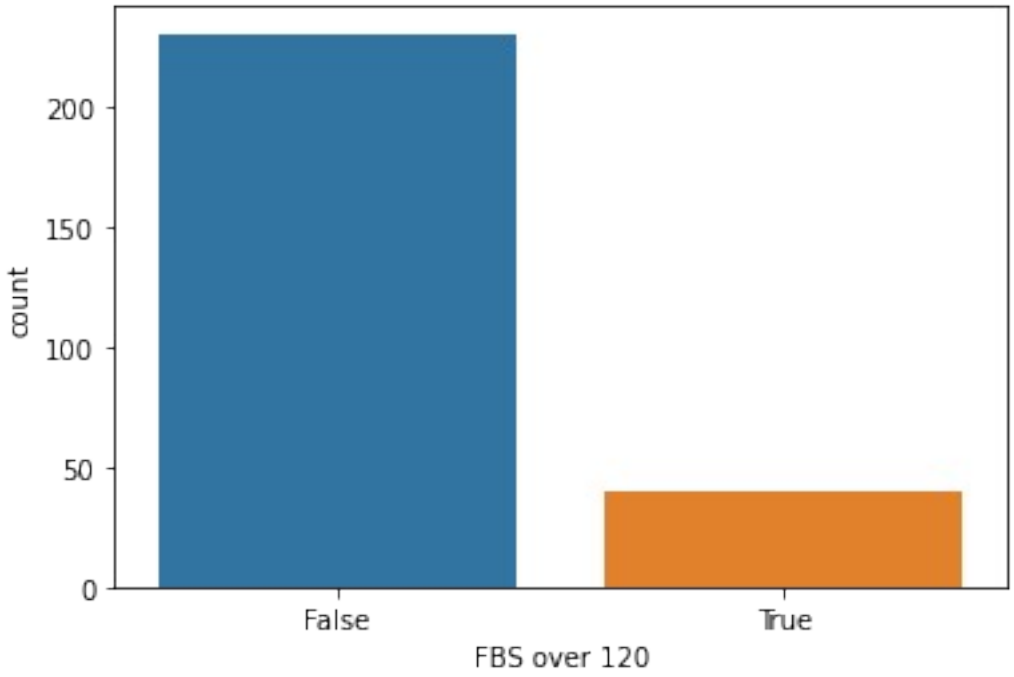
```
sns.histplot(x=data["Cholesterol"],kde=True)
plt.title("Cholesterol");
```




```
data['FBS over 120'].value_counts()
```

```
0    230
1     40
Name: FBS over 120, dtype: int64
```

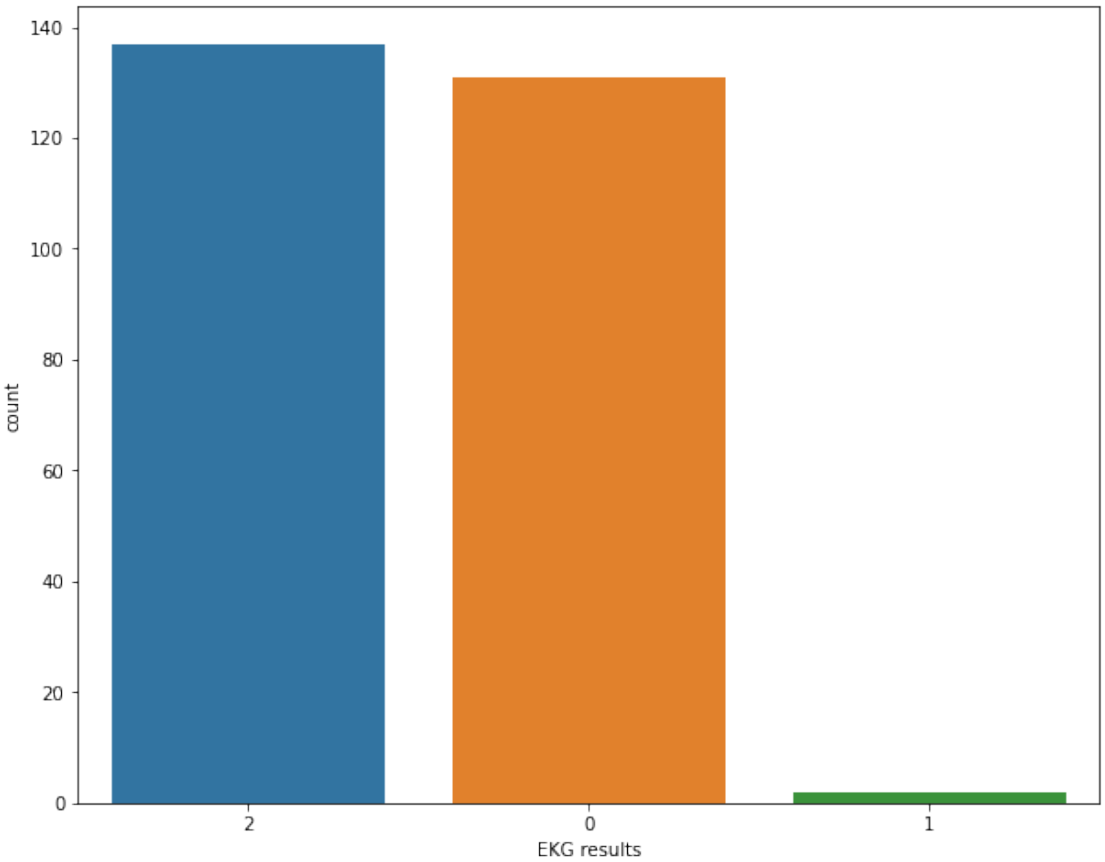
```
labels = ["False", 'True']
order = data['FBS over 120'].value_counts().index

sns.countplot(x='FBS over 120', data=data, order=order)
plt.xticks([0,1], labels=labels)
plt.show()
```



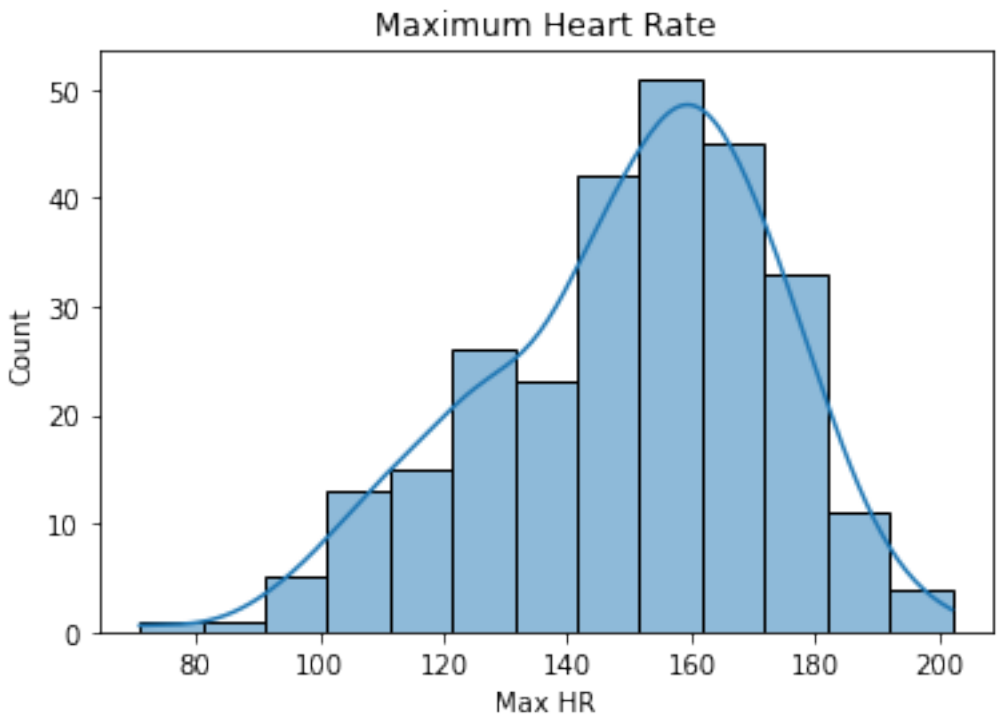

```
data['EKG results'].value_counts()
```

```
2    137
0    131
1      2
Name: EKG results, dtype: int64
```

```
labels = ["showing probable or definite left ventricular hypertrophy
by Estes' criteria", "normal",
          "having ST-T wave abnormality"]
order = data['EKG results'].value_counts().index

plt.figure(figsize=(10, 8))
sns.countplot(x='EKG results', data=data, order=order)
#plt.xticks([0,1,2], labels=labels)
plt.show()
```



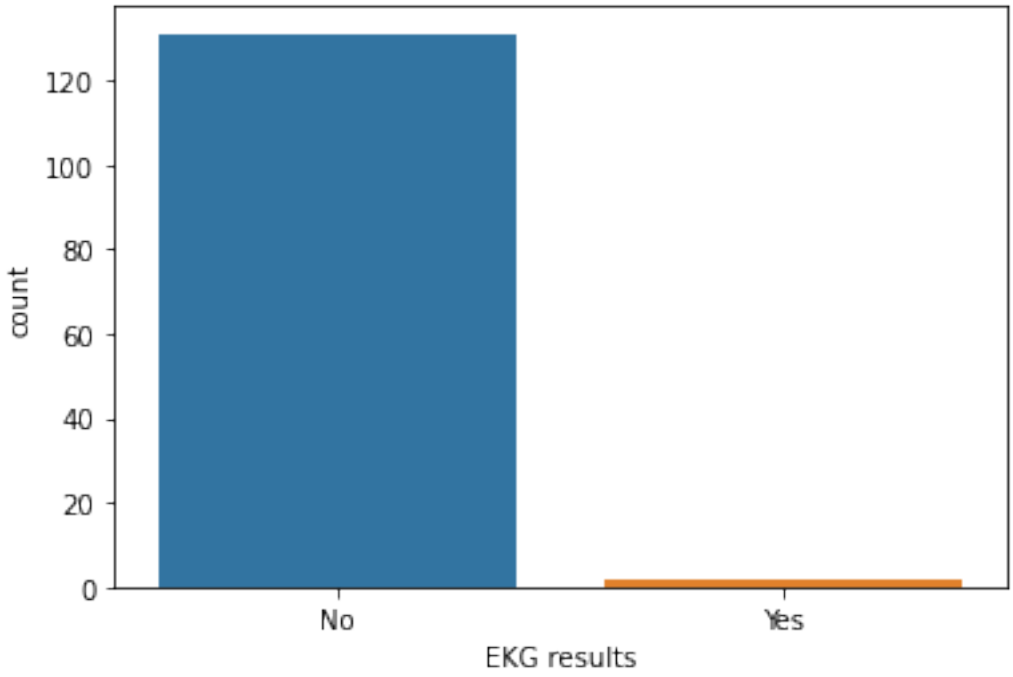

```
sns.histplot(x=data["Max HR"],kde=True)
plt.title("Maximum Heart Rate");
```



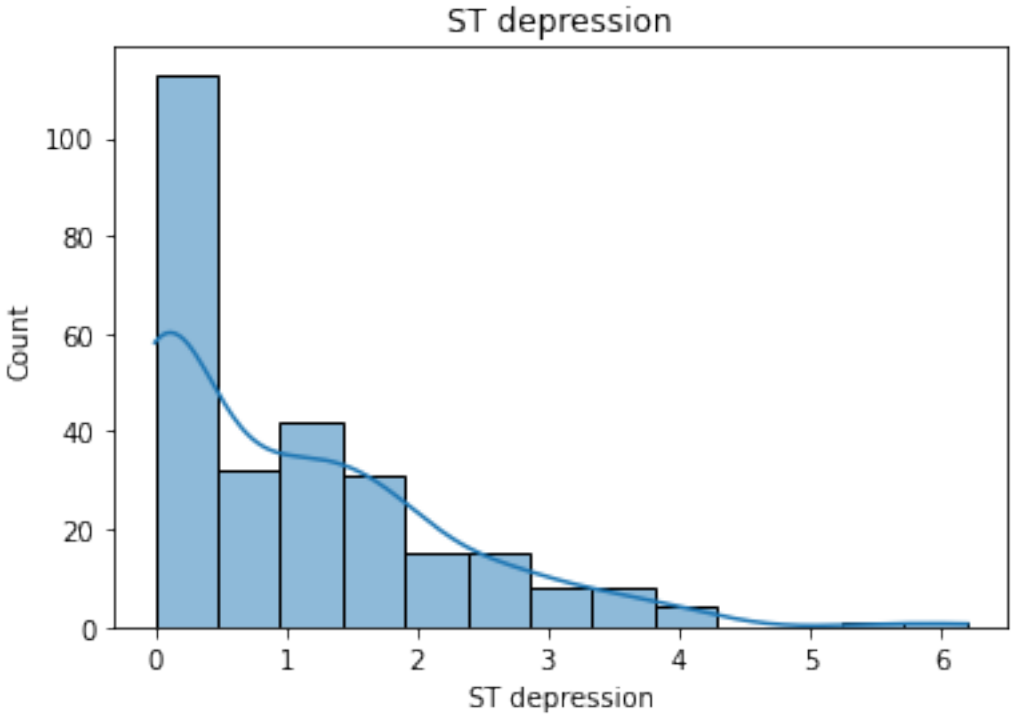

```
data['Exercise angina'].value_counts()
```

```
0    181
1     89
Name: Exercise angina, dtype: int64
```

```
labels = ["No", "Yes"]
order = data['Exercise angina'].value_counts().index

sns.countplot(x='EKG results', data=data, order=order)
plt.xticks([0,1], labels=labels)
plt.show()
```




```
sns.histplot(x=data["ST depression"],kde=True)
plt.title("ST depression");
```

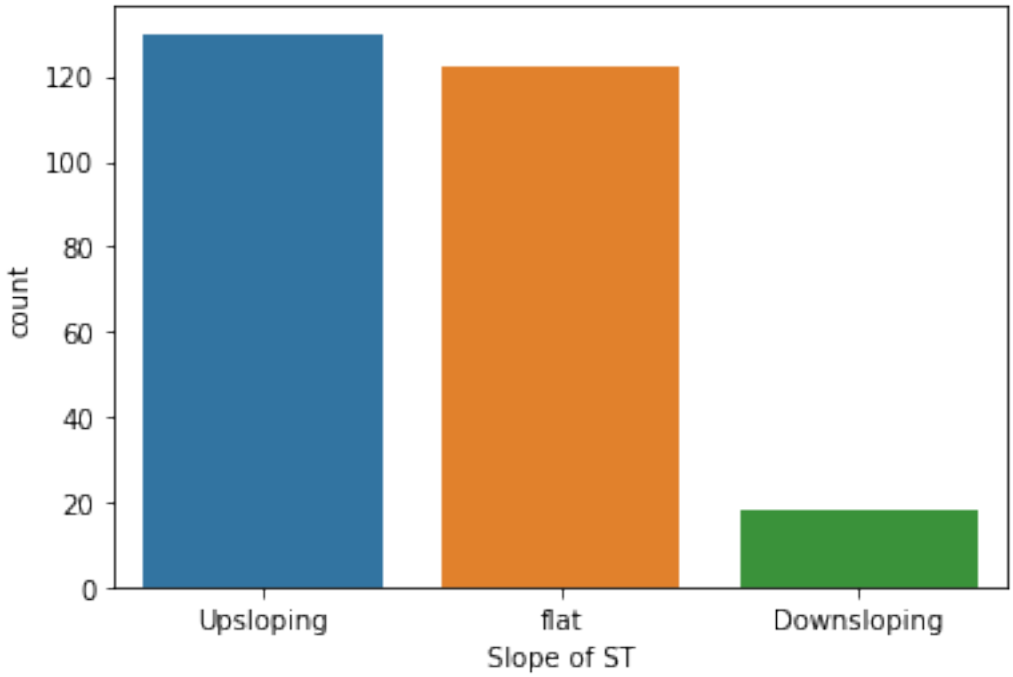



```
data['Slope of ST'].value_counts()
```

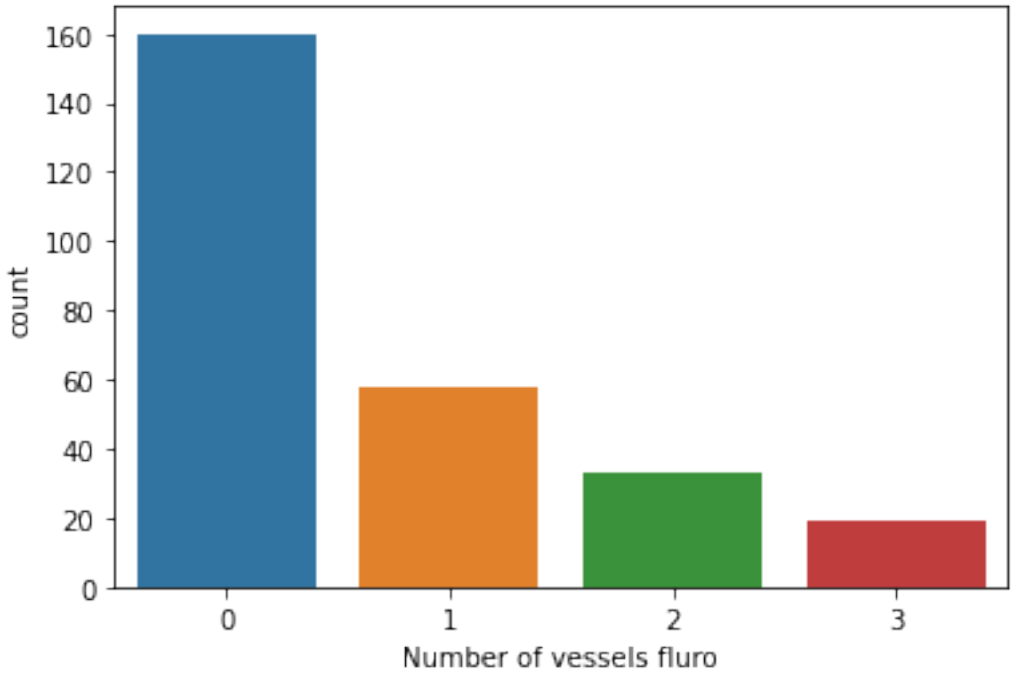
```
1    130
2    122
```

```
3     18
Name: Slope of ST, dtype: int64
```

```
labels = ["Upsloping", "flat","Downsloping"]
order = data['Slope of ST'].value_counts().index

sns.countplot(x='Slope of ST', data=data, order=order)
plt.xticks([0,1,2], labels=labels)
plt.show()
```



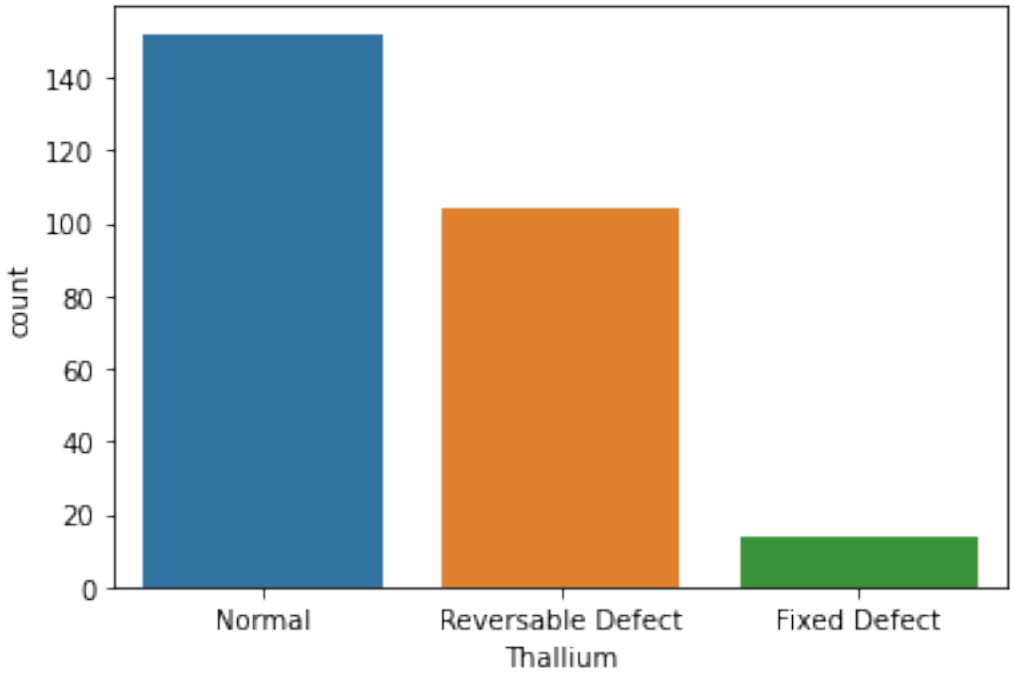

```
sns.countplot(x='Number of vessels fluro', data=data);
```




```
data['Thallium'].value_counts()
```

```
3    152
7    104
6     14
Name: Thallium, dtype: int64
```

```
labels = ["Normal","Reversable Defect","Fixed Defect"]
order = data['Thallium'].value_counts().index

sns.countplot(x='Thallium', data=data, order=order)
plt.xticks([0,1,2], labels=labels)
plt.show()
```

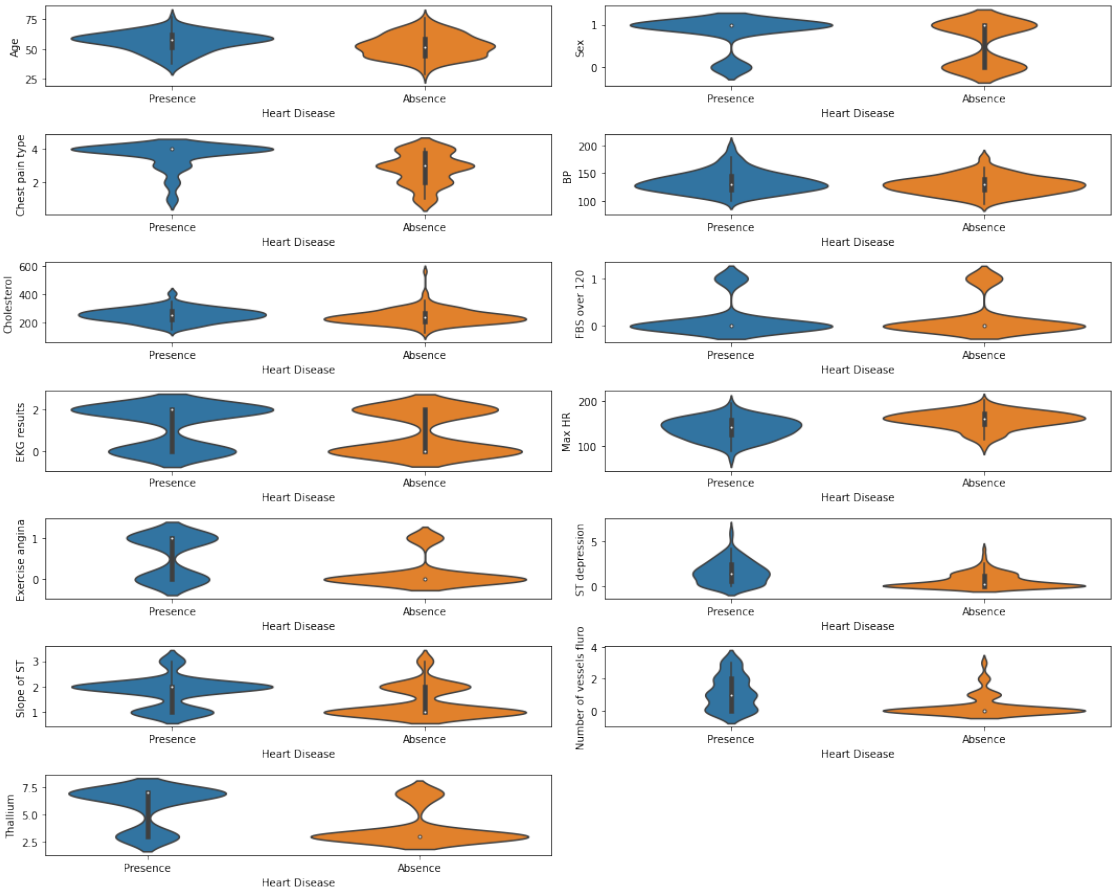



```
cols = ['Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol', 'FBS
over 120',
       'EKG results', 'Max HR', 'Exercise angina', 'ST depression',
       'Slope of ST', 'Number of vessels fluro', 'Thallium']
cols
```

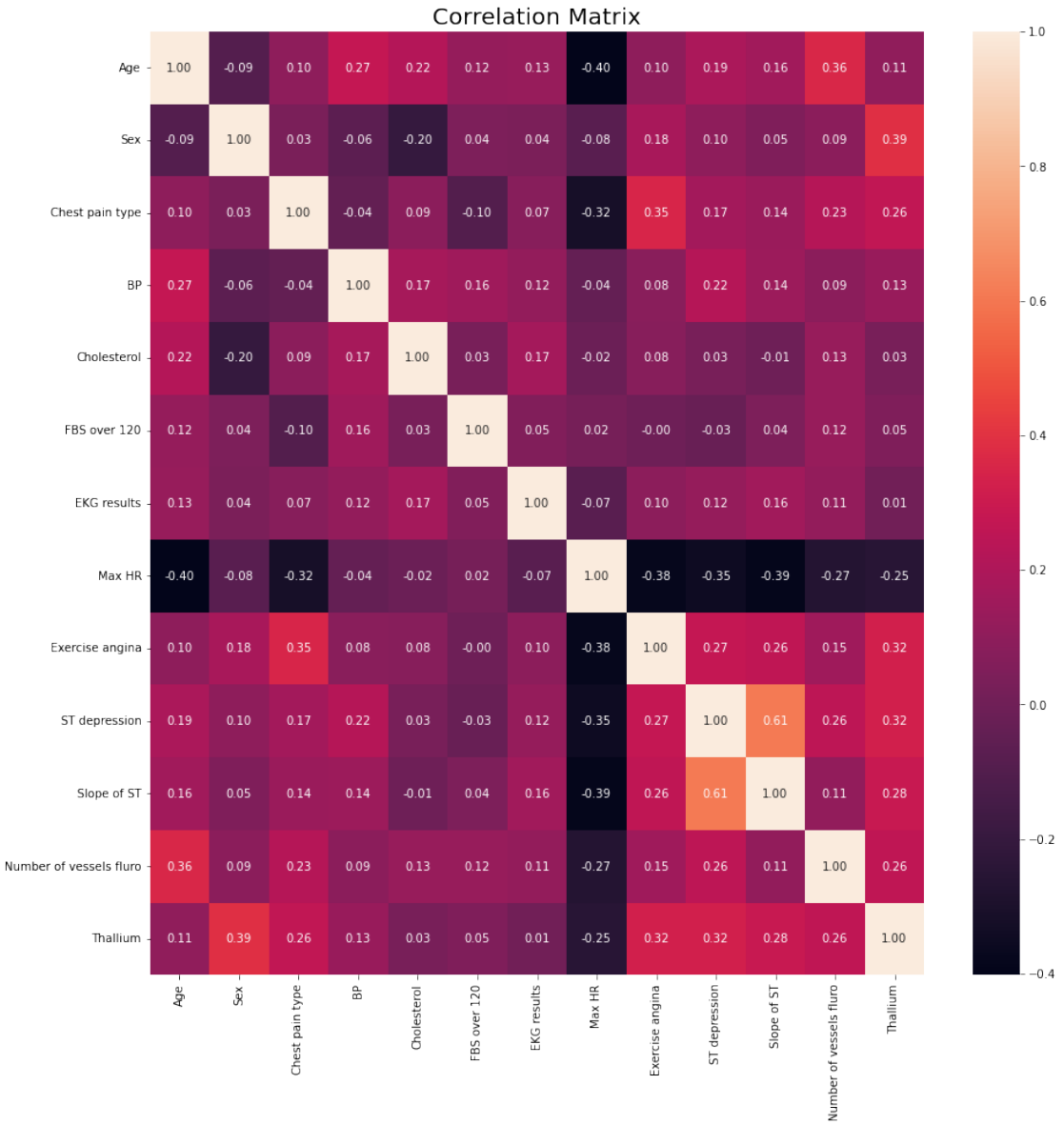
```
['Age',
 'Sex',
 'Chest pain type',
 'BP',
 'Cholesterol',
 'FBS over 120',
 'EKG results',
 'Max HR',
 'Exercise angina',
 'ST depression',
 'Slope of ST',
 'Number of vessels fluro',
 'Thallium']
```

```
plt.figure(figsize=(15, 12))
for i in range(len(cols)):
    plt.subplot(7, 2, i+1)
    sns.violinplot(x="Heart Disease", y=cols[i], data=data)
plt.tight_layout()
plt.plot()
```

```
[]
```




```
corr_matrix = data.corr()
plt.figure(figsize = (15, 15))
sns.heatmap(corr_matrix,annot=True,fmt='0.2f')
plt.title("Correlation Matrix", fontsize = 20)
plt.show()
```




```
data.columns
```

```
Index(['Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol', 'FBS over
120',
       'EKG results', 'Max HR', 'Exercise angina', 'ST depression',
       'Slope of ST', 'Number of vessels fluro', 'Thallium', 'Heart
Disease'],
      dtype='object')
```

```
x = data.iloc[:,0:13]
y = data['Heart Disease'].replace({"Absence":0,"Presence":1})
```

```
x.head()
```

```
   Age  Sex  Chest pain type   BP  Cholesterol  FBS over 120  EKG
results  \
```

```
0   70    1                4  130          322             0
2
1   67    0                3  115          564             0
2
2   57    1                2  124          261             0
0
3   64    1                4  128          263             0
0
4   74    0                2  120          269             0
2

   Max HR  Exercise angina  ST depression  Slope of ST  \
0     109                 0            2.4            2
1     160                 0            1.6            2
2     141                 0            0.3            1
3     105                 1            0.2            2
4     121                 1            0.2            1

   Number of vessels fluro  Thallium
0                        3         3
1                        0         7
2                        0         7
3                        1         7
4                        1         3

y

0      1
1      0
2      1
3      0
4      0
      ..
265    0
266    0
267    0
268    0
269    1
Name: Heart Disease, Length: 270, dtype: int64
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,
test_size=0.10)

print(f'x_train contains: {x_train.shape[0]} rows and
{x_train.shape[1]} columns')
print(f'x_test contains: {x_test.shape[0]} rows and {x_test.shape[1]}
columns')
```

```
x_train contains: 243 rows and 13 columns
x_test contains: 27 rows and 13 columns
```

```
print(f'y_train contains: {y_train.shape}')
print(f'y_test contains: {y_test.shape}')
```

```
y_train contains: (243,)
y_test contains: (27,)
```

```
print(f'x_train contains: {x_train.shape[0]} rows and
{x_train.shape[1]} columns')
print(f'x_test contains: {x_test.shape[0]} rows and {x_test.shape[1]}
columns')
```

```
x_train contains: 243 rows and 13 columns
x_test contains: 27 rows and 13 columns
```

```
print(f'y_train contains: {y_train.shape}')
print(f'y_test contains: {y_test.shape}')
```

```
y_train contains: (243,)
y_test contains: (27,)
```

```
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier()
```

```
knn.score(x_test,y_test)
```

```
0.5185185185185185
```

```
knn_pred = knn.predict(x_test)
```

```
len(knn_pred)
```

```
27
```

```
print(classification_report(y_test,knn_pred))
```

```
              precision    recall  f1-score   support

           0       0.71      0.53      0.61        19
           1       0.31      0.50      0.38         8

    accuracy                           0.52        27
   macro avg       0.51      0.51      0.49        27
weighted avg       0.59      0.52      0.54        27
```

```
print(classification_report(y_train, knn.predict(x_train)))
```

```
              precision    recall  f1-score   support

           0       0.77      0.85      0.81       131
           1       0.80      0.71      0.75       112

    accuracy                           0.78       243
   macro avg       0.78      0.78      0.78       243
weighted avg       0.78      0.78      0.78       243
```

```
x_test.shape
```

```
(27, 13)
```

```
from ibm_watson_machine_learning import APIClient
wml_credentials={
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "X2Pk1QhBMW2t1ZpNhVtmi3RShe8Icb-61H84qlBy3UyG"
}
client=APIClient(wml_credentials)
```

```
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    #print(space)
    return(next(item for item in space['resources'] if item['entity']
["name"] == space_name)['metadata']['id'])
```

```
space_uid = guid_from_space_name(client, 'models')
print("Space UID = " + space_uid)
```

```
Space UID = e74734cd-b84a-4c71-90d2-ceefc839e963
```

```
client.set.default_space(space_uid)
```

```
'SUCCESS'
```

```
client.software_specifications.list()
```

```
------------------------------  ------------------------------------
----
NAME                            ASSET_ID
TYPE
default_py3.6                   0062b8c9-8b7d-44a0-a9b9-46c416adcbd9
base
kernel-spark3.2-scala2.12       020d69ce-7ac1-5e68-ac1a-31189867356a
base
pytorch-onnx_1.3-py3.7-edt      069ea134-3346-5748-b513-49120e15d288
base
scikit-learn_0.20-py3.6         09c5a1d0-9c1e-4473-a344-eb7b665ff687
base
spark-mllib_3.0-scala_2.12      09f4cff0-90a7-5899-b9ed-1ef348aebdee
base
```

```
pytorch-onnx_rt22.1-py3.9      0b848dd4-e681-5599-be41-b5f6fccc6471
base
ai-function_0.1-py3.6          0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda
base
shiny-r3.6                     0e6e79df-875e-4f24-8ae9-62dcc2148306
base
tensorflow_2.4-py3.7-horovod   1092590a-307d-563d-9b62-4eb7d64b3f22
base
pytorch_1.1-py3.6              10ac12d6-6b30-4ccd-8392-3e922c096a92
base
tensorflow_1.15-py3.6-ddl      111e41b3-de2d-5422-a4d6-bf776828c4b7
base
autoai-kb_rt22.2-py3.10        125b6d9a-5b1f-5e8d-972a-b251688ccf40
base
runtime-22.1-py3.9             12b83a17-24d8-5082-900f-0ab31fbfd3cb
base
scikit-learn_0.22-py3.6        154010fa-5b3b-4ac1-82af-4d5ee5abbc85
base
default_r3.6                   1b70aec3-ab34-4b87-8aa0-a4a3c8296a36
base
pytorch-onnx_1.3-py3.6         1bc6029a-cc97-56da-b8e0-39c3880dbbe7
base
kernel-spark3.3-r3.6           1c9e5454-f216-59dd-a20e-474a5cdf5988
base
pytorch-onnx_rt22.1-py3.9-edt  1d362186-7ad5-5b59-8b6c-9d0880bde37f
base
tensorflow_2.1-py3.6           1eb25b84-d6ed-5dde-b6a5-3fbdf1665666
base
spark-mllib_3.2                20047f72-0a98-58c7-9ff5-a77b012eb8f5
base
tensorflow_2.4-py3.8-horovod   217c16f6-178f-56bf-824a-b19f20564c49
base
runtime-22.1-py3.9-cuda        26215f05-08c3-5a41-a1b0-da66306ce658
base
do_py3.8                       295addb5-9ef9-547e-9bf4-92ae3563e720
base
autoai-ts_3.8-py3.8            2aa0c932-798f-5ae9-abd6-15e0c2402fb5
base
tensorflow_1.15-py3.6          2b73a275-7cbf-420b-a912-eae7f436e0bc
base
kernel-spark3.3-py3.9          2b7961e2-e3b1-5a8c-a491-482c8368839a
base
pytorch_1.2-py3.6              2c8ef57d-2687-4b7d-acce-01f94976dac1
base
spark-mllib_2.3                2e51f700-bca0-4b0d-88dc-5c6791338875
base
pytorch-onnx_1.1-py3.6-edt     32983cea-3f32-4400-8965-dde874a8d67e
base
spark-mllib_3.0-py37           36507ebe-8770-55ba-ab2a-eafe787600e9
base
```

```
spark-mllib_2.4                390d21f8-e58b-4fac-9c55-d7ceda621326
base
autoai-ts_rt22.2-py3.10        396b2e83-0953-5b86-9a55-7ce1628a406f
base
xgboost_0.82-py3.6             39e31acd-5f30-41dc-ae44-60233c80306e
base
pytorch-onnx_1.2-py3.6-edt     40589d0e-7019-4e28-8daa-fb03b6f4fe12
base
pytorch-onnx_rt22.2-py3.10     40e73f55-783a-5535-b3fa-0c8b94291431
base
default_r36py38                41c247d3-45f8-5a71-b065-8580229facf0
base
autoai-ts_rt22.1-py3.9         4269d26e-07ba-5d40-8f66-2d495b0c71f7
base
autoai-obm_3.0                 42b92e18-d9ab-567f-988a-4240ba1ed5f7
base
pmml-3.0_4.3                   493bcb95-16f1-5bc5-bee8-81b8af80e9c7
base
spark-mllib_2.4-r_3.6          49403dff-92e9-4c87-a3d7-a42d0021c095
base
xgboost_0.90-py3.6             4ff8d6c2-1343-4c18-85e1-689c965304d3
base
pytorch-onnx_1.1-py3.6         50f95b2a-bc16-43bb-bc94-b0bed208c60b
base
autoai-ts_3.9-py3.8            52c57136-80fa-572e-8728-a5e7cbb42cde
base
spark-mllib_2.4-scala_2.11     55a70f99-7320-4be5-9fb9-9edb5a443af5
base
spark-mllib_3.0                5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9
base
autoai-obm_2.0                 5c2e37fa-80b8-5e77-840f-d912469614ee
base
spss-modeler_18.1              5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b
base
cuda-py3.8                     5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e
base
runtime-22.2-py3.10-xc         5e8cddff-db4a-5a6a-b8aa-2d4af9864dab
base
autoai-kb_3.1-py3.7            632d4b22-10aa-5180-88f0-f52dfb6444d7
base
----------------------------  ------------------------------------
----
Note: Only first 50 records were displayed. To display more use
'limit' parameter.
```

```
software_spec_uid =
client.software_specifications.get_uid_by_name("runtime-22.1-py3.9")
software_spec_uid
```

```
'12b83a17-24d8-5082-900f-0ab31fbfd3cb'
```