Health Care ML project

September 4, 2022

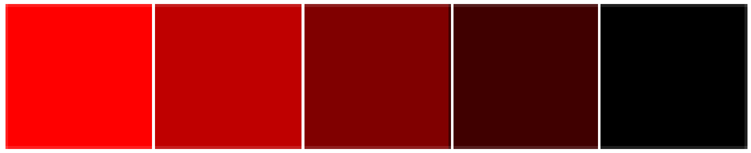
Introduction World Health Organization has estimated 12 million deaths occur worldwide, every year due to Heart diseases. Half the deaths in the United States and other developed countries are due to cardio vascular diseases. The early prognosis of cardiovascular diseases can aid in making decisions on lifestyle changes in high risk patients and in turn reduce the complications. In this notebook, the data given has the information about the factors that might have an impact on cardiovascular health. We have discovered each feature, took some insights and build logistic regression and random forest models in order to predict the risk of heart disease.

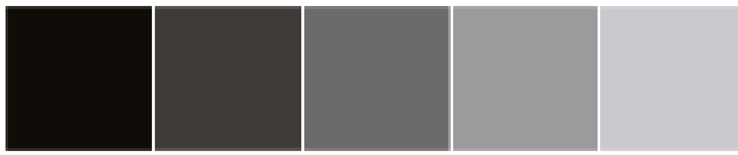
1. Importing Libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker
import seaborn as sns
from statsmodels.graphics.gofplots import qqplot
sns.set()
from operator import add
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score
%matplotlib inline

# Disable warnings
import warnings
warnings.filterwarnings("ignore")
```

2. Color Palettes

[2]:
```
# --- Create List of Color Palletes ---
red_grad = ['#FF0000', '#BF0000', '#800000', '#400000', '#000000']
black_grad = ['#100C07', '#3E3B39', '#6D6A6A', '#9B9A9C', '#CAC9CD']

# --- Plot Color Palletes --
sns.palplot(red_grad)
sns.palplot(black_grad)
```





3. Reading Dataset After importing libraries, the dataset that will be used will be imported.

[3]:
```
#load the data
data = pd.read_csv('1645792390_cep1_dataset.csv')
data.head()
```

[3]:
```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   3       145   233    1        0      150      0      2.3      0
1   37    1   2       130   250    0        1      187      0      3.5      0
2   41    0   1       130   204    0        0      172      0      1.4      2
3   56    1   1       120   236    0        1      178      0      0.8      2
4   57    0   0       120   354    0        1      163      1      0.6      2

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
```

[4]:
```
data.columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',␣
 ↪'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']
```

[5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
```

```
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

4. EDA

[6]:
```python
#total percentage of missing data
missing_data = data.isnull().sum()
total_percentage = (missing_data.sum()/data.shape[0]) * 100
print(f'The total percentage of missing data is {round(total_percentage,2)}%')
```

```
The total percentage of missing data is 0.0%
```

Luckily, there is no missing values There are 1 float64 columns and 6 int64/float64 columns.

[7]:
```python
print(data.duplicated())
```

```
0      False
1      False
2      False
3      False
4      False
       …
298    False
299    False
300    False
301    False
302    False
Length: 303, dtype: bool
```

[8]:
```python
y = data.drop_duplicates()
print(y)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
```

```
0    63   1   3      145  233   1        0       150       0        2.3
1    37   1   2      130  250   0        1       187       0        3.5
2    41   0   1      130  204   0        0       172       0        1.4
3    56   1   1      120  236   0        1       178       0        0.8
4    57   0   0      120  354   0        1       163       1        0.6
..   …  …  ..       …   …  …      …      …     …     …
298  57   0   0      140  241   0        1       123       1        0.2
299  45   1   3      110  264   0        1       132       0        1.2
300  68   1   0      144  193   1        1       141       0        3.4
301  57   1   0      130  131   0        1       115       1        1.2
302  57   0   1      130  236   0        0       174       0        0.0

     slope  ca  thal  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     …  ..   …     …
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[302 rows x 14 columns]
```

We had only one duplicated row

It's a clean, easy to understand set of data. However, the meaning of some of the column headers are not obvious. Here's what they mean:

age: The person's age in years

sex: The person's sex (1 = male, 0 = female)

cp: The chest pain experienced (Value 0: typical angina, Value 1: atypical angina, Value 2: non-anginal pain, Value 3: asymptomatic)

trestbps: The person's resting blood pressure (mm Hg on admission to the hospital)

chol: The person's cholesterol measurement in mg/dl

fbs: The person's fasting blood sugar ($>$ 120 mg/dl, 1 = true; 0 = false)

restecg: Resting electrocardiographic measurement (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria)

thalachu: The person's maximum heart rate achieved

exang: Exercise induced angina (1 = yes; 0 = no)

oldpeak: ST depression induced by exercise relative to rest ('ST' relates to positions on the ECG plot.)

slope: the slope of the peak exercise ST segment (Value 1: upsloping, Value 2: flat, Value 3: downsloping)

ca: The number of major vessels (0-3)

thal: A blood disorder called thalassemia (0 = no thalassemia; 1 = normal, 2 = fixed defect; 3 = reversable defect)

target: Heart disease (0 = no, 1 = yes)

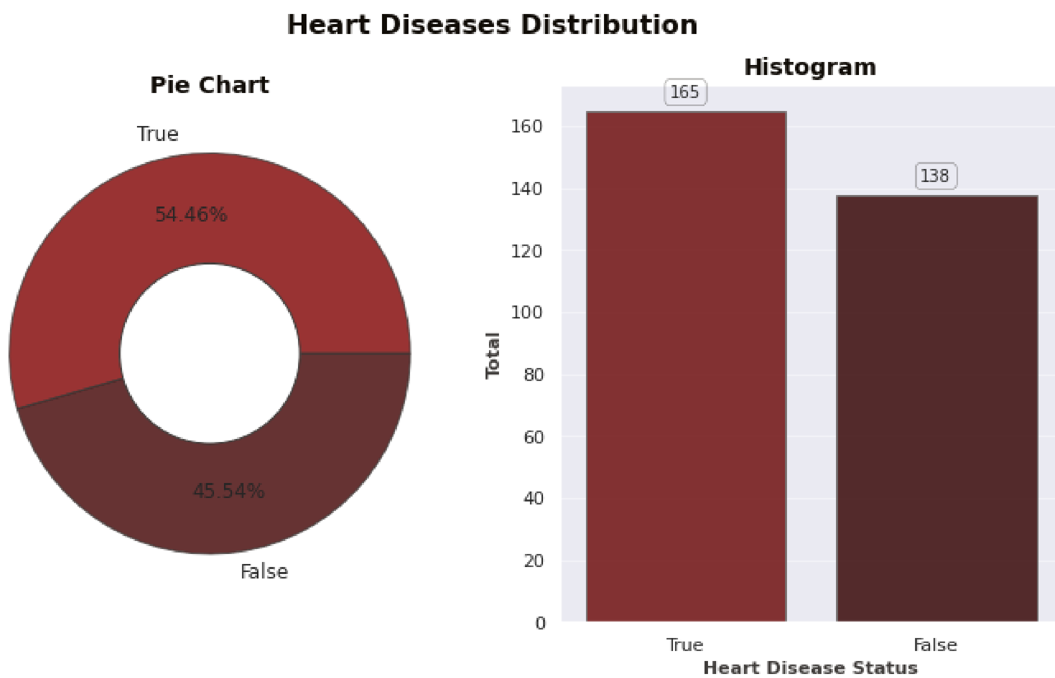
5.Initial Data Exploration First, analysing the target variable

[9]:
```
# --- Setting Colors, Labels, Order ---
colors=red_grad[2:5]
labels=['True', 'False']
order=data['target'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(12,6))
plt.suptitle('Heart Diseases Distribution', fontweight='heavy',
             fontsize=16, fontfamily='sans-serif', color=black_grad[0])

# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
plt.pie(data['target'].value_counts(), labels=labels, colors=colors,
        wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]), autopct='%.2f%%',
        pctdistance=0.7, textprops={'fontsize':12})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
plt.gcf().gca().add_artist(centre)

# --- Histogram ---
countplt = plt.subplot(1, 2, 2)
plt.title('Histogram', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
ax = sns.countplot(x='target', data=data, palette=colors, order=order,
                   edgecolor=black_grad[2], alpha=0.85)
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2,
             rect.get_height()+4.25,rect.get_height(),
             horizontalalignment='center', fontsize=10,
             bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.
↪25,
                       boxstyle='round'))

plt.xlabel('Heart Disease Status', fontweight='bold', fontsize=11,
           fontfamily='sans-serif', color=black_grad[1])
```

```
plt.ylabel('Total', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.xticks([0, 1], labels)
plt.grid(axis='y', alpha=0.4)
countplt

# --- Count Categorical Labels w/out Dropping Null Walues ---
print('*' * 45)
print('\033[1m'+'.: Heart Diseases Status (target) Total :.'+'\033[0m')
print('*' * 45)
data.target.value_counts(dropna=False)
```

```
*********************************************
.: Heart Diseases Status (target) Total :.
*********************************************
```

[9]:
```
1    165
0    138
Name: target, dtype: int64
```



The percentage of patients with heart problems is 54.46%.

6. Peliminary Statistical Summary of the Data

[10]:
```
data.describe()
```

[10]:
```
              age         sex          cp    trestbps        chol         fbs  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean    54.366337    0.683168    0.966997  131.623762  246.264026    0.148515
std      9.082101    0.466011    1.032052   17.538143   51.830751    0.356198
min     29.000000    0.000000    0.000000   94.000000  126.000000    0.000000
25%     47.500000    0.000000    0.000000  120.000000  211.000000    0.000000
50%     55.000000    1.000000    1.000000  130.000000  240.000000    0.000000
75%     61.000000    1.000000    2.000000  140.000000  274.500000    0.000000
max     77.000000    1.000000    3.000000  200.000000  564.000000    1.000000

          restecg     thalach       exang     oldpeak       slope          ca  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean     0.528053  149.646865    0.326733    1.039604    1.399340    0.729373
std      0.525860   22.905161    0.469794    1.161075    0.616226    1.022606
min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000  133.500000    0.000000    0.000000    1.000000    0.000000
50%      1.000000  153.000000    0.000000    0.800000    1.000000    0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

             thal      target
count  303.000000  303.000000
mean     2.313531    0.544554
std      0.612277    0.498835
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

[11]:
```
data.median()
```

[11]:
```
age          55.0
sex           1.0
cp            1.0
trestbps    130.0
chol        240.0
fbs           0.0
restecg       1.0
thalach     153.0
exang         0.0
oldpeak       0.8
slope         1.0
ca            0.0
thal          2.0
target        1.0
dtype: float64
```

[12]:
```
data.mode()
```

[12]:
```
    age  sex   cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0  58.0  1.0  0.0     120.0   197  0.0      1.0    162.0    0.0      0.0
1   NaN  NaN  NaN       NaN   204  NaN      NaN      NaN    NaN      NaN
2   NaN  NaN  NaN       NaN   234  NaN      NaN      NaN    NaN      NaN

   slope   ca  thal  target
0    2.0  0.0   2.0     1.0
1    NaN  NaN   NaN     NaN
2    NaN  NaN   NaN     NaN
```

[13]:
```
data["target"].describe()
```

[13]:
```
count    303.000000
mean       0.544554
std        0.498835
min        0.000000
25%        0.000000
50%        1.000000
75%        1.000000
max        1.000000
Name: target, dtype: float64
```

7.Data Exploration

We'll analyse 'sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca' and 'thal' features

Analysing the 'Sex' feature

[14]:
```
data["sex"].unique()
```

[14]:
```
array([1, 0])
```

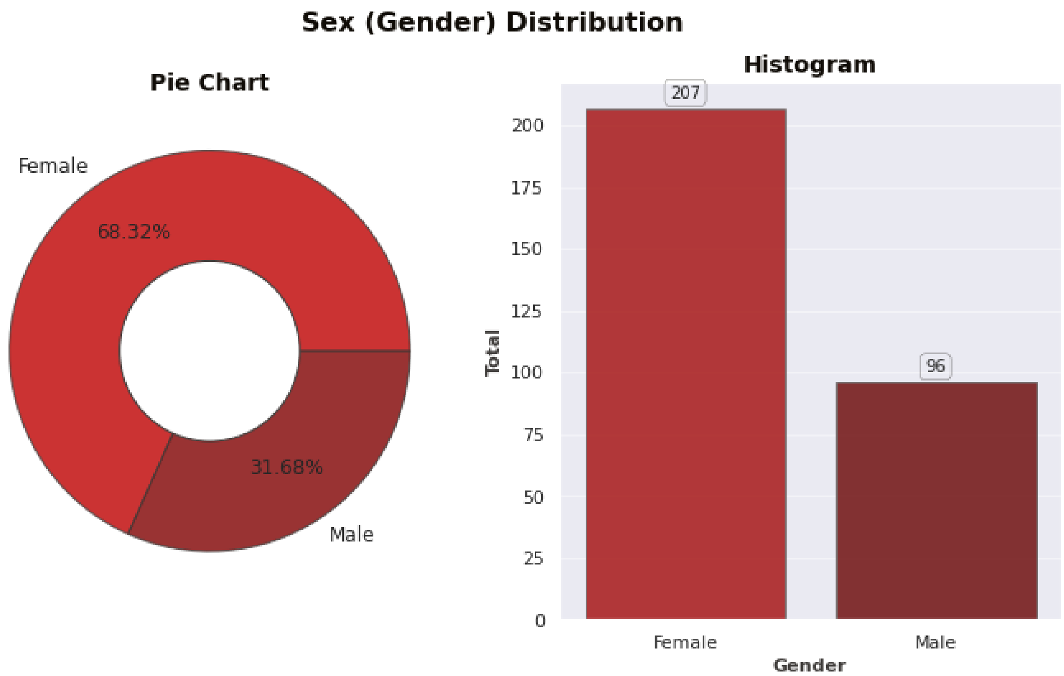
0 is for female and 1 for male

[15]:
```
# --- Setting Colors, Labels, Order ---
colors=red_grad[1:4]
labels=['Female', 'Male']
order=data['sex'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(12, 6))
plt.suptitle('Sex (Gender) Distribution', fontweight='heavy',
             fontsize='16', fontfamily='sans-serif', color=black_grad[0])

# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontweight='bold', fontsize=14,
```

```
          fontfamily='sans-serif', color=black_grad[0])
plt.pie(data['sex'].value_counts(), labels=labels, colors=colors, pctdistance=0.
 ↪7,
        autopct='%.2f%%', wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]),
        textprops={'fontsize':12})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
plt.gcf().gca().add_artist(centre)

# --- Histogram ---
countplt = plt.subplot(1, 2, 2)
plt.title('Histogram', fontweight='bold', fontsize=14,
          fontfamily='sans-serif', color=black_grad[0])
ax = sns.countplot(x='sex', data=data, palette=colors, order=order,
                   edgecolor=black_grad[2], alpha=0.85)
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2,
             rect.get_height()+4.25,rect.get_height(),
             horizontalalignment='center', fontsize=10,
             bbox=dict(facecolor='none', edgecolor=black_grad[0],
                       linewidth=0.25, boxstyle='round'))

plt.xlabel('Gender', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.xticks([0, 1], labels)
plt.grid(axis='y', alpha=0.4)
countplt
# --- Count Categorical Labels w/out Dropping Null Walues ---
print('*' * 25)
print('\033[1m'+'.: Sex (Gender) Total :.'+'\033[0m')
print('*' * 25)
data.sex.value_counts(dropna=False)
```

```
*************************
.: Sex (Gender) Total :.
*************************
```

[15]:
```
1    207
0     96
Name: sex, dtype: int64
```



There are significantly more women in the data than men

Analysing the 'Chest Pain Type' feature

[16]:
```
data["cp"].unique()
```

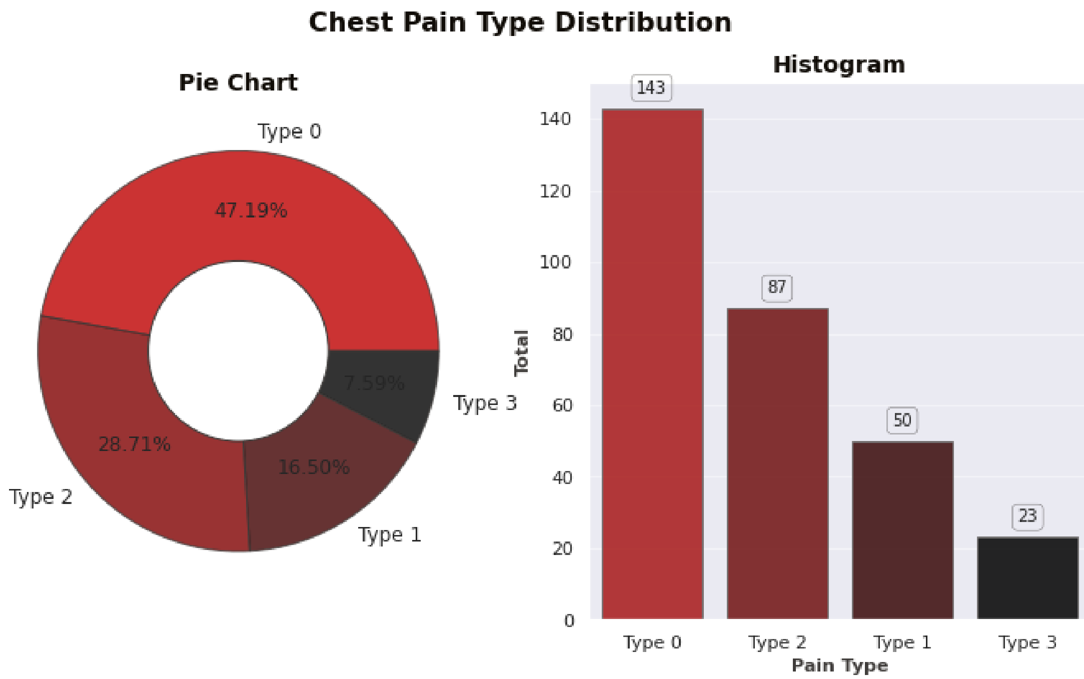
[16]:
```
array([3, 2, 1, 0])
```

[17]:
```
# --- Setting Colors, Labels, Order ---
colors=red_grad[1:5]
labels=['Type 0', 'Type 2', 'Type 1', 'Type 3']
order=data['cp'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(12, 6))
plt.suptitle('Chest Pain Type Distribution', fontweight='heavy', fontsize=16,
             fontfamily='sans-serif', color=black_grad[0])

# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontweight='bold', fontsize=14,fontfamily='sans-serif',
          color=black_grad[0])
plt.pie(data['cp'].value_counts(), labels=labels, colors=colors, pctdistance=0.
 ↪7,
        autopct='%.2f%%', textprops={'fontsize':12},
        wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]))
```

```python
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
plt.gcf().gca().add_artist(centre)

# --- Histogram ---
countplt = plt.subplot(1, 2, 2)
plt.title('Histogram', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
ax = sns.countplot(x='cp', data=data, palette=colors, order=order,
                   edgecolor=black_grad[2], alpha=0.85)
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2,
             rect.get_height()+4.25,rect.get_height(),
             horizontalalignment='center', fontsize=10,
             bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.
 ↪25,
                       boxstyle='round'))

plt.xlabel('Pain Type', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.xticks([0, 1, 2, 3], labels)
plt.grid(axis='y', alpha=0.4)
countplt

# --- Count Categorical Labels w/out Dropping Null Walues ---
print('*' * 30)
print('\033[1m'+'.: Chest Pain Type Total :.'+'\033[0m')
print('*' * 30)
data.cp.value_counts(dropna=False)
```

```
******************************
.: Chest Pain Type Total :.
******************************
```

[17]:
```
0    143
2     87
1     50
3     23
Name: cp, dtype: int64
```



We notice, that chest pain of '0' is most frequent in the data.

Analysing the FBS feature

[18]:
```
data["fbs"].unique()
```

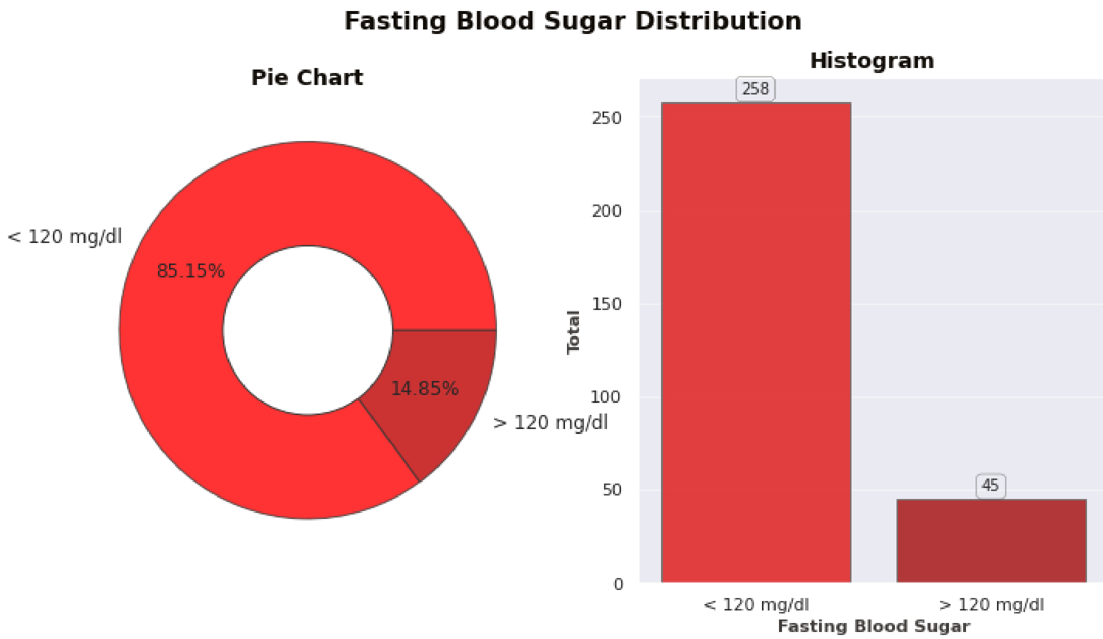
[18]:
```
array([1, 0])
```

[19]:
```
# --- Setting Colors, Labels, Order ---
colors=red_grad[0:5]
labels=['< 120 mg/dl', '> 120 mg/dl']
order=data['fbs'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(12, 6))
plt.suptitle('Fasting Blood Sugar Distribution', fontweight='heavy',
             fontsize=16, fontfamily='sans-serif', color=black_grad[0])

# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
plt.pie(data['fbs'].value_counts(), labels=labels, colors=colors,
        wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]), autopct='%.2f%%',
        pctdistance=0.7, textprops={'fontsize':12})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
```

```
plt.gcf().gca().add_artist(centre)

# --- Histogram ---
countplt = plt.subplot(1, 2, 2)
plt.title('Histogram', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
ax = sns.countplot(x='fbs', data=data, palette=colors, order=order,
                   edgecolor=black_grad[2], alpha=0.85)
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2,
             rect.get_height()+4.25,rect.get_height(),
             horizontalalignment='center', fontsize=10,
             bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.
 ↪25,
                       boxstyle='round'))

plt.xlabel('Fasting Blood Sugar', fontweight='bold', fontsize=11,
           fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.xticks([0, 1], labels)
plt.grid(axis='y', alpha=0.4)
countplt


# --- Count Categorical Labels w/out Dropping Null Walues ---
print('*' * 32)
print('\033[1m'+'.: Fasting Blood Sugar Total :.'+'\033[0m')
print('*' * 32)
data.fbs.value_counts(dropna=False)
```

```
********************************
.: Fasting Blood Sugar Total :.
********************************
```

[19]:
```
0    258
1     45
Name: fbs, dtype: int64
```



Nothing extraordinary here

Analysing the restecg feature

[20]:
```
data["restecg"].unique()
```

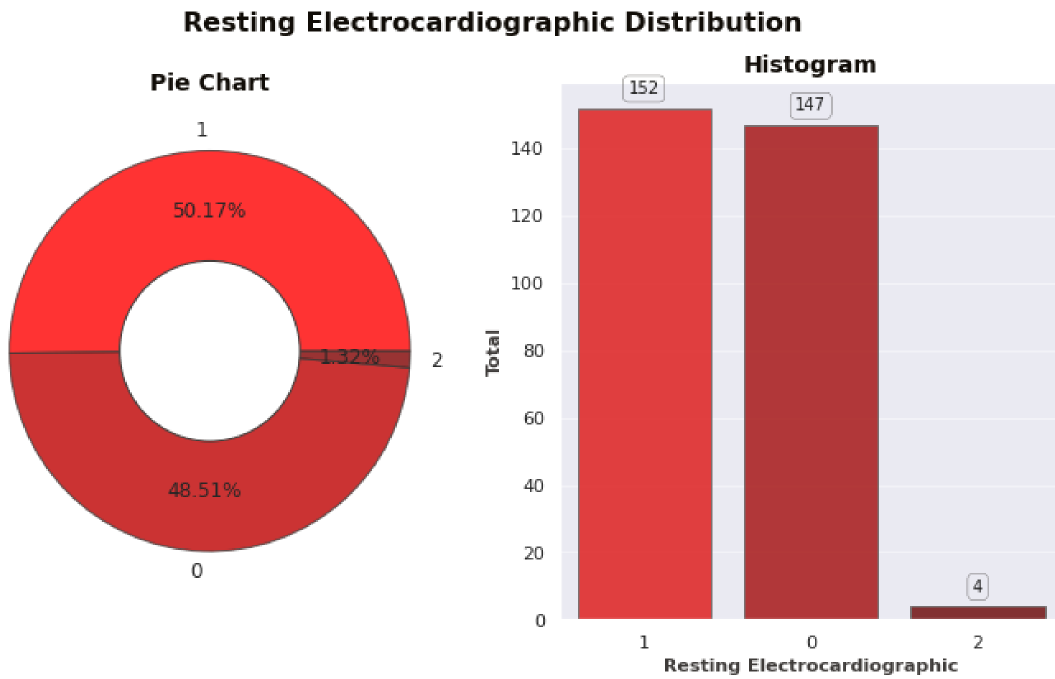
[20]:
```
array([0, 1, 2])
```

[21]:
```
# --- Setting Colors, Labels, Order ---
colors=red_grad[0:5]
labels=['1', '0', '2']
order=data['restecg'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(12, 6))
plt.suptitle('Resting Electrocardiographic Distribution', fontweight='heavy',
             fontsize=16, fontfamily='sans-serif', color=black_grad[0])

# --- Pie Chart ---
plt.subplot(1,2,1)
plt.title('Pie Chart', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
plt.pie(data['restecg'].value_counts(), labels=labels, colors=colors,
        wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]), autopct='%.2f%%',
        pctdistance=0.7, textprops={'fontsize':12})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
plt.gcf().gca().add_artist(centre)
```

```
# --- Histogram ---
countplt = plt.subplot(1, 2, 2)
plt.title('Histogram', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
ax = sns.countplot(x='restecg', data=data, palette=colors, order=order,
                   edgecolor=black_grad[2], alpha=0.85)
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2,
             rect.get_height()+4.25,rect.get_height(),
             horizontalalignment='center', fontsize=10,
             bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.
↪25,
                       boxstyle='round'))

plt.xlabel('Resting Electrocardiographic', fontweight='bold', fontsize=11,
           fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.grid(axis='y', alpha=0.4)
countplt

# --- Count Categorical Labels w/out Dropping Null Walues ---
print('*' * 50)
print('\033[1m'+'.: Resting Electrocardiographic Results Total :.'+'\033[0m')
print('*' * 50)
data.restecg.value_counts(dropna=False)
```

```
**************************************************
.: Resting Electrocardiographic Results Total :.
**************************************************
```

[21]:
```
1    152
0    147
2      4
Name: restecg, dtype: int64
```



Most of the patients has restecg '1' or '0'.

[22]:
```
data["exang"].unique()
```

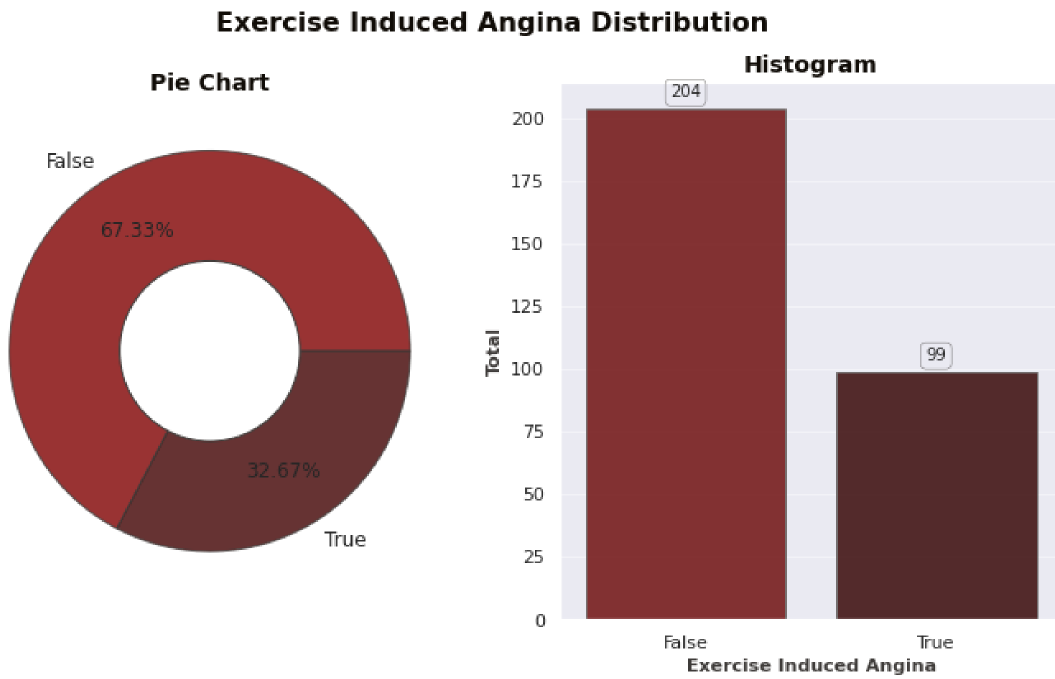
[22]: array([0, 1])

[23]:
```
# --- Setting Colors, Labels, Order ---
colors=red_grad[2:5]
labels=['False', 'True']
order=data['exang'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(12, 6))
plt.suptitle('Exercise Induced Angina Distribution', fontweight='heavy',
             fontsize=16, fontfamily='sans-serif', color=black_grad[0])

# --- Pie Chart ---
plt.subplot(1,2,1)
plt.title('Pie Chart', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
plt.pie(data['exang'].value_counts(), labels=labels, colors=colors,
        wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]), autopct='%.2f%%',
        pctdistance=0.7, textprops={'fontsize':12})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
plt.gcf().gca().add_artist(centre)
```

```
# --- Histogram ---
countplt = plt.subplot(1, 2, 2)
plt.title('Histogram', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
ax = sns.countplot(x='exang', data=data, palette=colors, order=order,
                   edgecolor=black_grad[2], alpha=0.85)
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2,
             rect.get_height()+4.25,rect.get_height(),
             horizontalalignment='center', fontsize=10,
             bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.
↪25,
                       boxstyle='round'))

plt.xlabel('Exercise Induced Angina', fontweight='bold', fontsize=11,
           fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.xticks([0, 1], labels)
plt.grid(axis='y', alpha=0.4)
countplt

# --- Count Categorical Labels w/out Dropping Null Walues ---
print('*' * 35)
print('\033[1m'+'.: Exercise Induced Angina Total :.'+'\033[0m')
print('*' * 35)
data.exang.value_counts(dropna=False)
```

```
***********************************
.: Exercise Induced Angina Total :.
***********************************
```

[23]:
```
0    204
1     99
Name: exang, dtype: int64
```



Analysing the Slope feature

[24]:
```
data["slope"].unique()
```

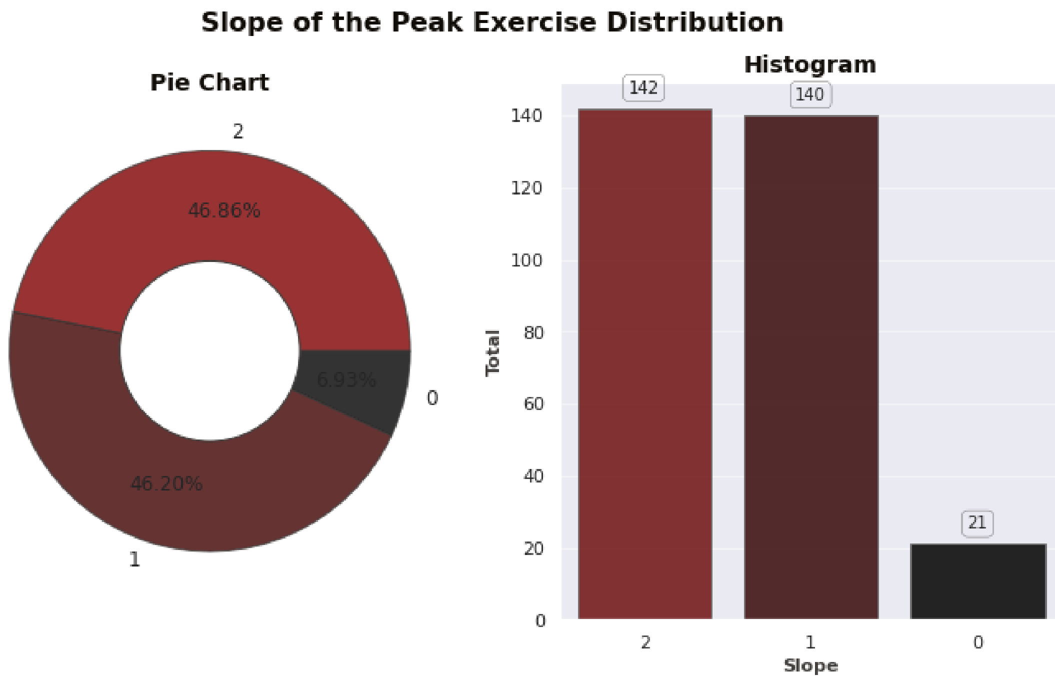
[24]: array([0, 2, 1])

[25]:
```
# --- Setting Colors, Labels, Order ---
colors=red_grad[2:5]
labels=['2', '1', '0']
order=data['slope'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(12, 6))
plt.suptitle('Slope of the Peak Exercise Distribution', fontweight='heavy',
             fontsize=16, fontfamily='sans-serif', color=black_grad[0])

# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontweight='bold', fontsize=14,
          fontfamily='sans-serif', color=black_grad[0])
plt.pie(data['slope'].value_counts(), labels=labels, colors=colors,
        wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]), autopct='%.2f%%',
        pctdistance=0.7, textprops={'fontsize':12})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
plt.gcf().gca().add_artist(centre)
```

```
# --- Histogram ---
countplt = plt.subplot(1, 2, 2)
plt.title('Histogram', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
ax = sns.countplot(x='slope', data=data
                   , palette=colors, order=order,
                   edgecolor=black_grad[2], alpha=0.85)
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2,
             rect.get_height()+4.25,rect.get_height(),
             horizontalalignment='center', fontsize=10,
             bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.
 ↪25,
                       boxstyle='round'))

plt.xlabel('Slope', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.grid(axis='y', alpha=0.4)
countplt
# --- Count Categorical Labels w/out Dropping Null Walues ---
print('*' * 20)
print('\033[1m'+'.: Slope Total :.'+'\033[0m')
print('*' * 20)
data.slope.value_counts(dropna=False)
```

```
********************
.: Slope Total :.
********************
```

[25]:
```
2    142
1    140
0     21
Name: slope, dtype: int64
```



Analysing the 'ca' feature

[26]:
```
data["ca"].unique()
```

[26]: `array([0, 2, 1, 3, 4])`

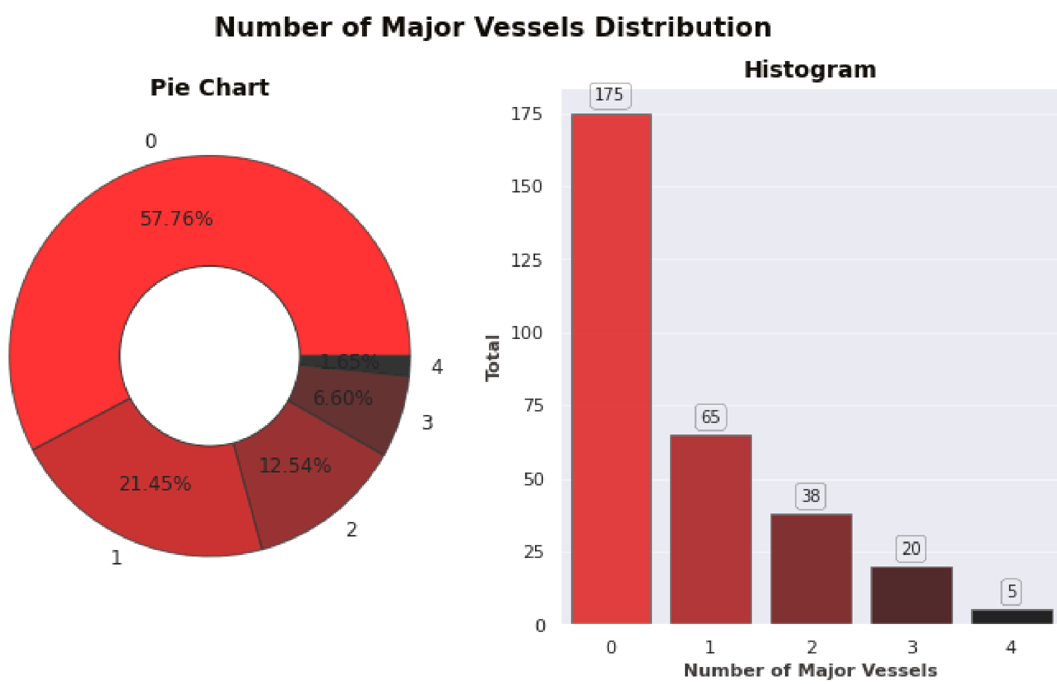
[27]:
```
# --- Setting Colors, Labels, Order ---
colors=red_grad
labels=['0', '1', '2', '3', '4']
order=data['ca'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(12, 6))
plt.suptitle('Number of Major Vessels Distribution', fontweight='heavy',
             fontsize=16, fontfamily='sans-serif', color=black_grad[0])

# --- Pie Chart ---
plt.subplot(1,2,1)
plt.title('Pie Chart', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
plt.pie(data['ca'].value_counts(), labels=labels, colors=colors,
        wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]),
        autopct='%.2f%%', pctdistance=0.7, textprops={'fontsize':12})

centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
```

```
plt.gcf().gca().add_artist(centre)

# --- Histogram ---
countplt = plt.subplot(1, 2, 2)
plt.title('Histogram', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
ax = sns.countplot(x='ca', data=data, palette=colors, order=order,
                   edgecolor=black_grad[2], alpha=0.85)
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2,
             rect.get_height()+4.25,rect.get_height(),
             horizontalalignment='center', fontsize=10,
             bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.
 ↪25,
                       boxstyle='round'))

plt.xlabel('Number of Major Vessels', fontweight='bold', fontsize=11,
           fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.grid(axis='y', alpha=0.4)
countplt

# --- Count Categorical Labels w/out Dropping Null Walues ---
print('*' * 40)
print('\033[1m'+'.: Number of Major Vessels Total :.'+'\033[0m')
print('*' * 40)
data.ca.value_counts(dropna=False)
```

```
****************************************
.: Number of Major Vessels Total :.
****************************************
```

[27]:

```
0    175
1     65
2     38
3     20
4      5
Name: ca, dtype: int64
```



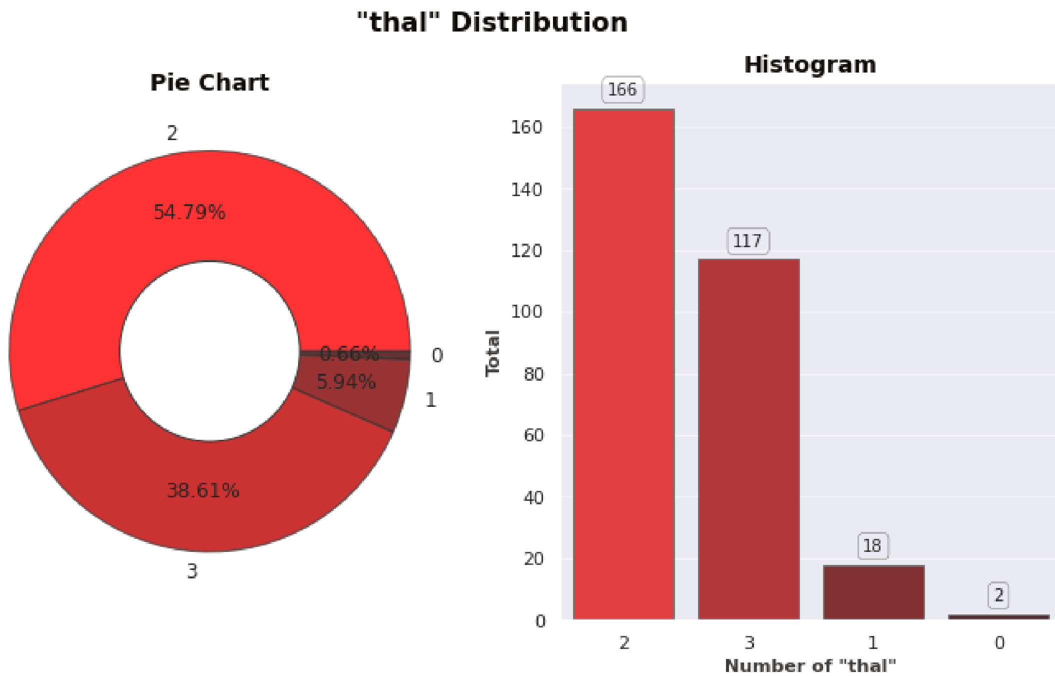
Analysing the 'thal' feature

[28]:
```
# --- Setting Colors, Labels, Order ---
colors=red_grad[0:4]
labels=['2', '3', '1', '0']
order=data['thal'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(12,6))
plt.suptitle('"thal" Distribution', fontweight='heavy', fontsize=16,
             fontfamily='sans-serif', color=black_grad[0])

# --- Pie Chart ---
plt.subplot(1,2,1)
plt.title('Pie Chart', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
plt.pie(data['thal'].value_counts(), labels=labels, colors=colors,
        wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]),
        autopct='%.2f%%', pctdistance=0.7, textprops={'fontsize':12})

centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
plt.gcf().gca().add_artist(centre)

# --- Histogram ---
countplt = plt.subplot(1, 2, 2)
```

```
plt.title('Histogram', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[0])
ax = sns.countplot(x='thal', data=data, palette=colors, order=order,
                   edgecolor=black_grad[2], alpha=0.85)
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2,
             rect.get_height()+4.25,rect.get_height(),
             horizontalalignment='center', fontsize=10,
             bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.
 ↪25,
                       boxstyle='round'))

plt.xlabel('Number of "thal"', fontweight='bold', fontsize=11,
           fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])
plt.grid(axis='y', alpha=0.4)
countplt

# --- Count Categorical Labels w/out Dropping Null Walues ---
print('*' * 20)
print('\033[1m'+'.: "thal" Total :.'+'\033[0m')
print('*' * 20)
data.thal.value_counts(dropna=False)
```

```
********************
.: "thal" Total :.
********************
```

[28]:
```
2    166
3    117
1     18
0      2
Name: thal, dtype: int64
```



8. EDA This section will perform some EDA to get more insights about dataset.

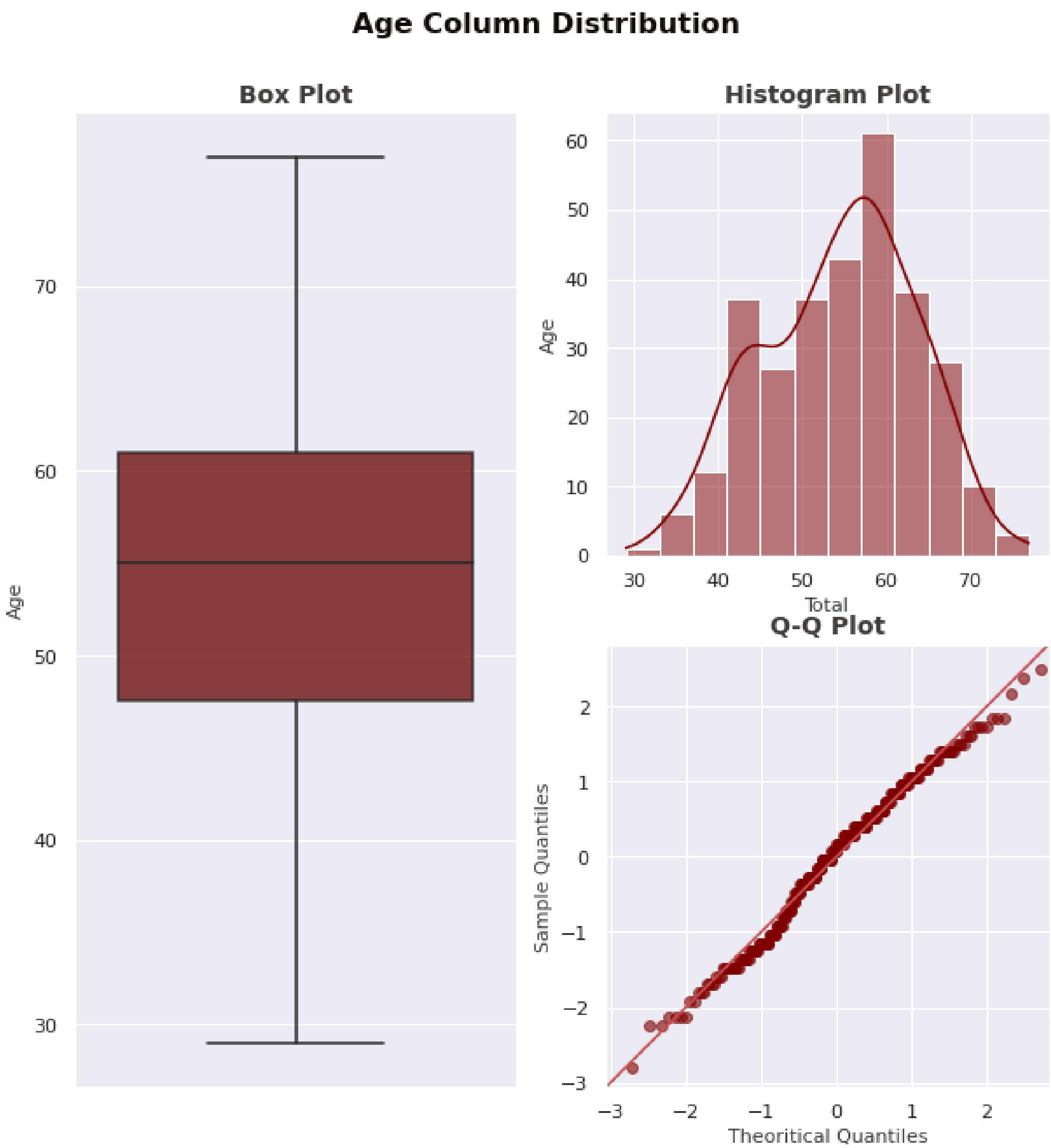
Let's check the occurrence of CVD across the Age category

First, we will check the distribution of ages

[29]:
```
# --- Variable, Color & Plot Size ---
var = 'age'
color = red_grad[2]
fig=plt.figure(figsize=(10, 10))

# --- Skewness & Kurtosis ---
print('\033[1m'+'.: Age Column Skewness & Kurtosis :.'+'\033[0m')
print('*' * 40)
print('Skewness:'+'\033[1m {:.3f}'.format(data[var].skew(axis = 0, skipna =␣
↪True)))
print('\033[0m'+'Kurtosis:'+'\033[1m {:.3f}'.format(data[var].kurt(axis = 0,␣
↪skipna = True)))
print('\n')

# --- General Title ---
fig.suptitle('Age Column Distribution', fontweight='bold', fontsize=16,
             fontfamily='sans-serif', color=black_grad[0])
fig.subplots_adjust(top=0.9)
```

```
# --- Histogram ---
ax_1=fig.add_subplot(2, 2, 2)
plt.title('Histogram Plot', fontweight='bold', fontsize=14,
          fontfamily='sans-serif', color=black_grad[1])
sns.histplot(data=data, x=var, kde=True, color=color)
plt.xlabel('Total', fontweight='regular', fontsize=11,
           fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Age', fontweight='regular', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])

# --- Q-Q Plot ---
ax_2=fig.add_subplot(2, 2, 4)
plt.title('Q-Q Plot', fontweight='bold', fontsize=14,
          fontfamily='sans-serif', color=black_grad[1])
qqplot(data[var], fit=True, line='45', ax=ax_2, markerfacecolor=color,
       markeredgecolor=color, alpha=0.6)
plt.xlabel('Theoritical Quantiles', fontweight='regular', fontsize=11,
           fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Sample Quantiles', fontweight='regular', fontsize=11,
           fontfamily='sans-serif', color=black_grad[1])

# --- Box Plot ---
ax_3=fig.add_subplot(1, 2, 1)
plt.title('Box Plot', fontweight='bold', fontsize=14, fontfamily='sans-serif',
          color=black_grad[1])
sns.boxplot(data=data, y=var, color=color, boxprops=dict(alpha=0.8),␣
 ↪linewidth=1.5)
plt.ylabel('Age', fontweight='regular', fontsize=11, fontfamily='sans-serif',
           color=black_grad[1])

plt.show()
```

```
.: Age Column Skewness & Kurtosis :.
****************************************
Skewness: -0.202

Kurtosis: -0.542
```

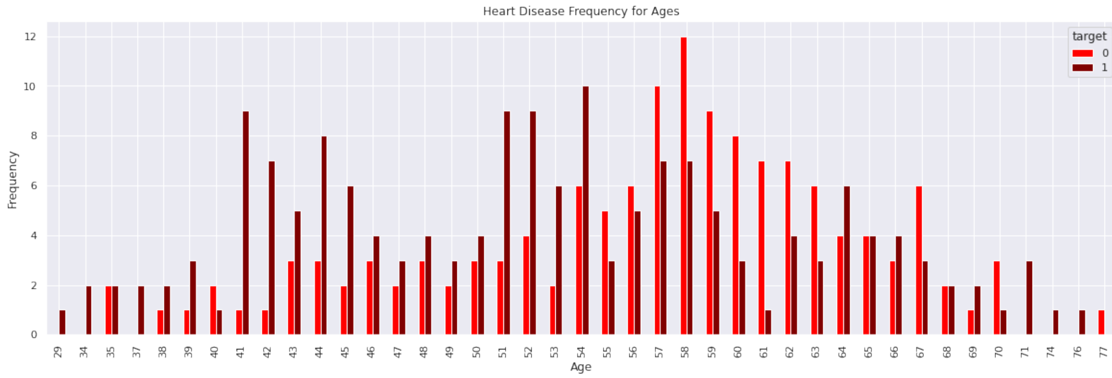


Most of people in this data are between 41 and 67 years old.

We notice that there are very few young people in this data, and they indeed have higher rates of disease. This clearly contradicts common sense and is most likely due to selection bias in the study.

[30]:
```
pd.crosstab(data.age,data.target).plot(kind="bar",figsize=(20,6), color=␣
↪['#FF0000','#800000'])
plt.title('Heart Disease Frequency for Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.savefig('heartDiseaseAndAges.png')
```

```
plt.show()
```



The patient in the data distributed from the age of 29 into 77. Age with the highest number with heart disease is 54.

We notice that there are very few young people in this data, and they indeed have higher rates of disease. This clearly contradicts common sense and is most likely due to selection bias in the study.

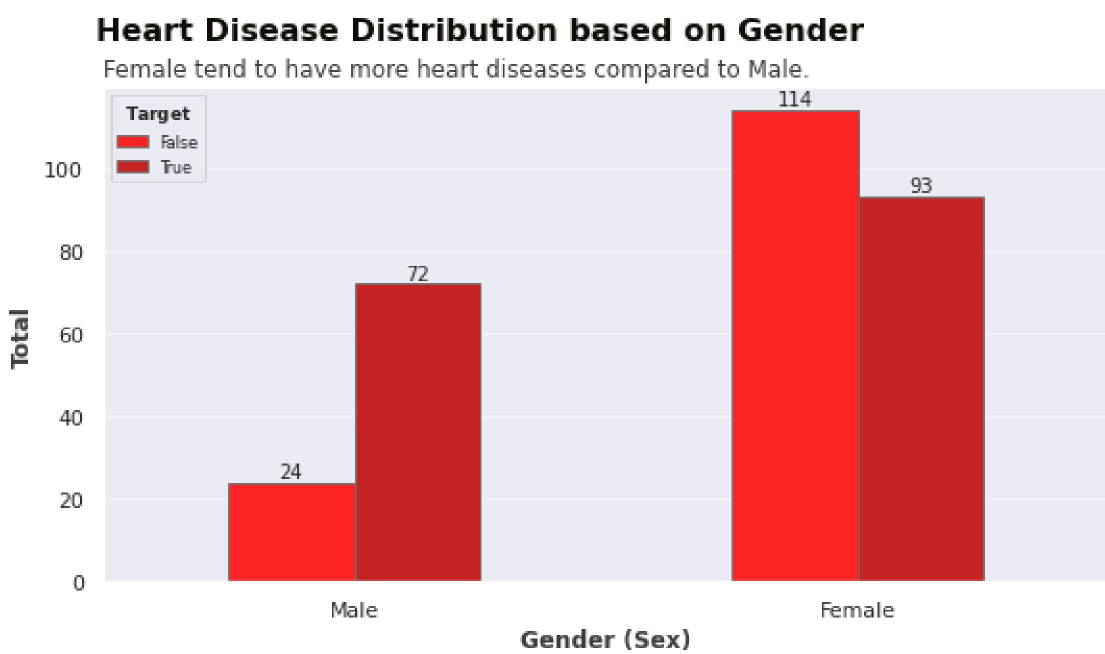
Gender As we noticed before we have significantly more women in the data than men

[31]:
```
data.groupby("sex") [["age", 'cp', 'trestbps', 'chol', 'fbs', 'restecg',␣
↪'thalach',
        'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']].mean()
```

[31]:
```
           age        cp    trestbps        chol      fbs   restecg  \
sex
0    55.677083  1.041667  133.083333  261.302083  0.12500  0.572917
1    53.758454  0.932367  130.946860  239.289855  0.15942  0.507246

        thalach     exang   oldpeak     slope        ca      thal    target
sex
0    151.125000  0.229167  0.876042  1.427083  0.552083  2.125000  0.750000
1    148.961353  0.371981  1.115459  1.386473  0.811594  2.400966  0.449275
```

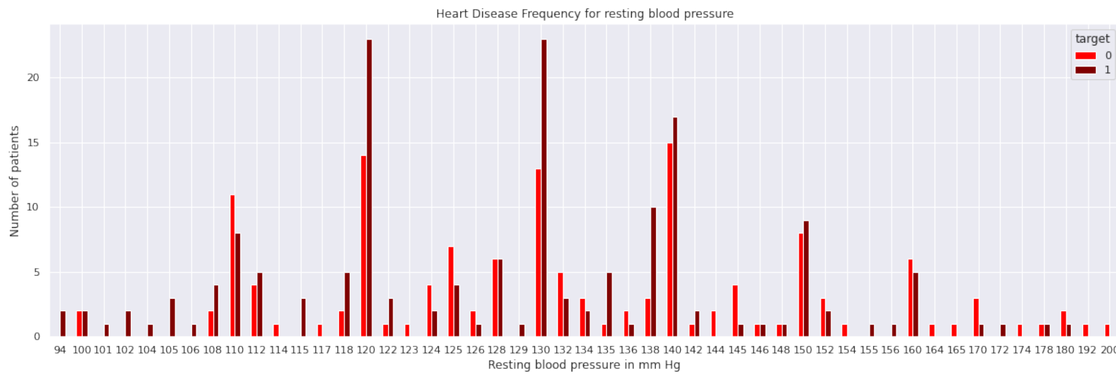
[32]:
```
# --- Labels Settings ---
labels = ['False', 'True']
label_gender = np.array([0, 1])
label_gender2 = ['Male', 'Female']

# --- Creating Bar Chart ---
ax = pd.crosstab(data.sex, data.target).plot(kind='bar', figsize=(8, 5),
                                              color=red_grad[0:4],
                                              edgecolor=black_grad[2], alpha=0.85)
```

```python
# --- Bar Chart Settings ---
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2,
             rect.get_height()+1.25,rect.get_height(),
             horizontalalignment='center', fontsize=10)

plt.suptitle('Heart Disease Distribution based on Gender', fontweight='heavy',
             x=0.065, y=0.98, ha='left', fontsize='16', fontfamily='sans-serif',
             color=black_grad[0])
plt.title('Female tend to have more heart diseases compared to Male.',
          fontsize='12', fontfamily='sans-serif', loc='left',␣
 ↪color=black_grad[1])
plt.tight_layout(rect=[0, 0.04, 1, 1.025])
plt.xlabel('Gender (Sex)', fontfamily='sans-serif', fontweight='bold',
           color=black_grad[1])
plt.ylabel('Total', fontfamily='sans-serif', fontweight='bold',
           color=black_grad[1])
plt.xticks(label_gender, label_gender2, rotation=0)
plt.grid(axis='y', alpha=0.4)
plt.grid(axis='x', alpha=0)
plt.legend(labels=labels, title='$\\bf{Target}$', fontsize='8',
           title_fontsize='9', loc='upper left', frameon=True);
```



[33]:
```python
pd.crosstab(data.trestbps,data.target).
 ↪plot(kind="bar",figsize=(20,6),color=['#FF0000', '#800000'])
```

```
plt.title('Heart Disease Frequency for resting blood pressure')
plt.xlabel('Resting blood pressure in mm Hg')
plt.xticks(rotation = 0)
plt.ylabel('Number of patients')
plt.show()
```

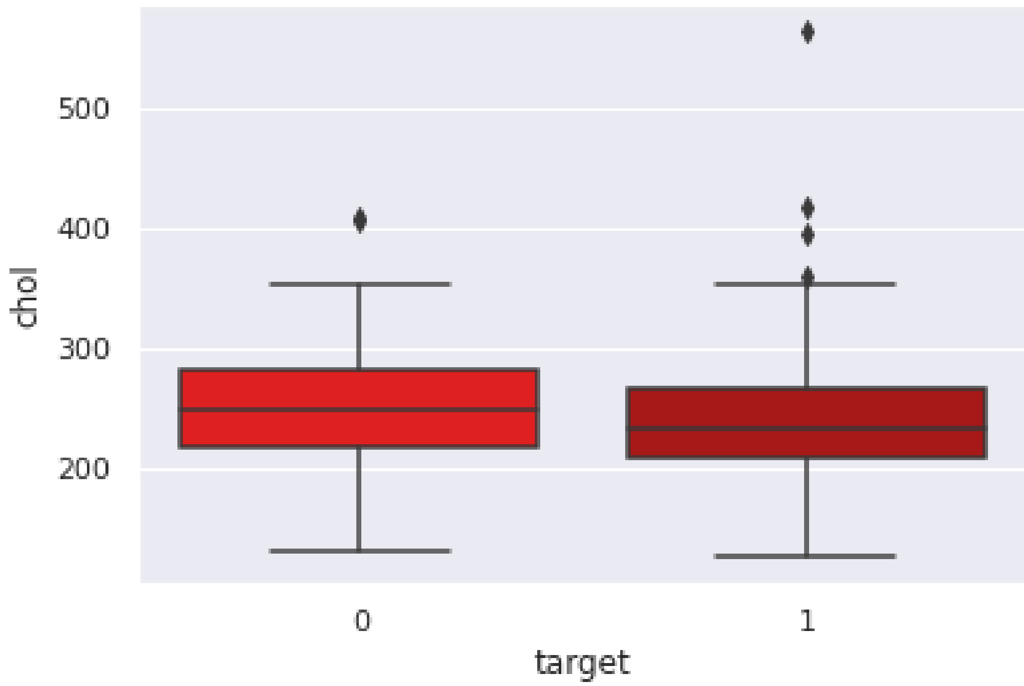


[34]:
```
data["chol"].unique()
```

[34]:
```
array([233, 250, 204, 236, 354, 192, 294, 263, 199, 168, 239, 275, 266,
       211, 283, 219, 340, 226, 247, 234, 243, 302, 212, 175, 417, 197,
       198, 177, 273, 213, 304, 232, 269, 360, 308, 245, 208, 264, 321,
       325, 235, 257, 216, 256, 231, 141, 252, 201, 222, 260, 182, 303,
       265, 309, 186, 203, 183, 220, 209, 258, 227, 261, 221, 205, 240,
       318, 298, 564, 277, 214, 248, 255, 207, 223, 288, 160, 394, 315,
       246, 244, 270, 195, 196, 254, 126, 313, 262, 215, 193, 271, 268,
       267, 210, 295, 306, 178, 242, 180, 228, 149, 278, 253, 342, 157,
       286, 229, 284, 224, 206, 167, 230, 335, 276, 353, 225, 330, 290,
       172, 305, 188, 282, 185, 326, 274, 164, 307, 249, 341, 407, 217,
       174, 281, 289, 322, 299, 300, 293, 184, 409, 259, 200, 327, 237,
       218, 319, 166, 311, 169, 187, 176, 241, 131])
```

[35]:
```
sns.boxplot(data = data, x = 'target', y = 'chol', palette= red_grad)
```

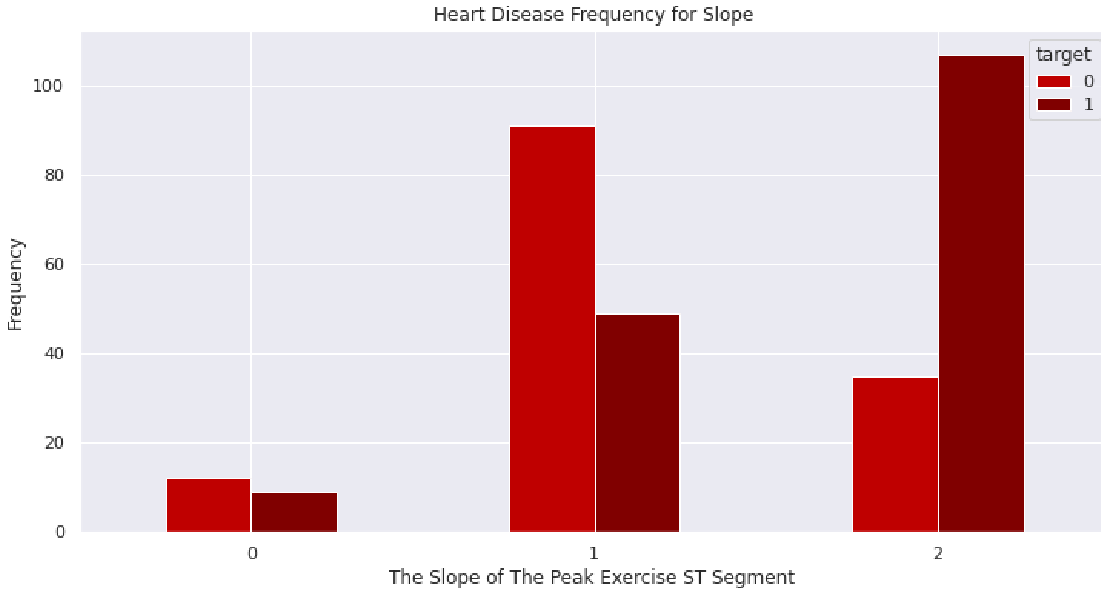
[35]: <AxesSubplot:xlabel='target', ylabel='chol'>



[36]:
```
data["slope"].unique()
```

[36]: array([0, 2, 1])

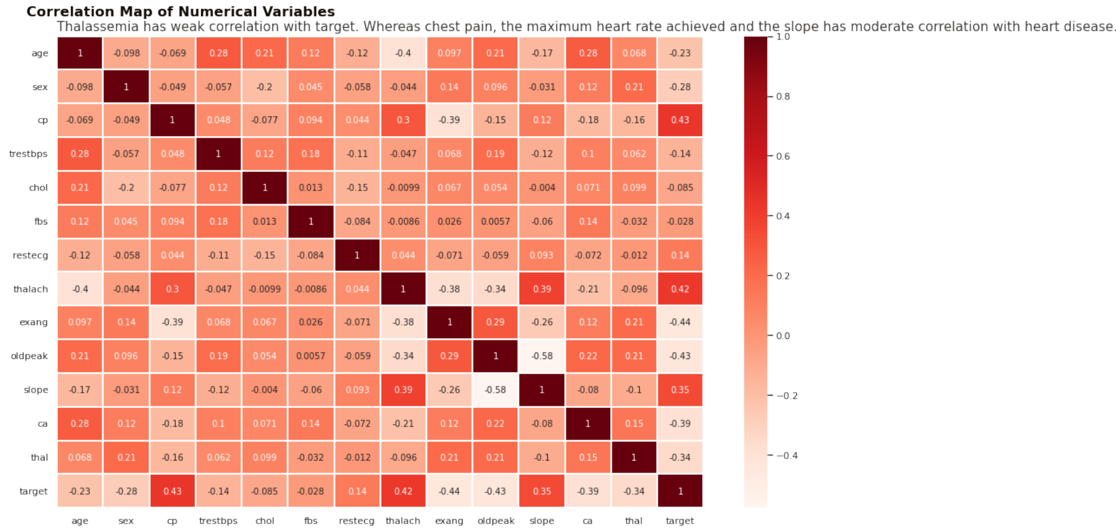
[37]:
```
pd.crosstab(data.slope,data.target).
 ↪plot(kind="bar",figsize=(12,6),color=['#BF0000', '#800000'])
plt.title('Heart Disease Frequency for Slope')
plt.xlabel('The Slope of The Peak Exercise ST Segment ')
plt.xticks(rotation = 0)
plt.ylabel('Frequency')
plt.show()
```



Heart disease occurs most when the slope of the peak exercise is 1

Heatmap Below is correlation map/heatmap of numerical variables to show correlation level/values for each variables with others

[38]:
```
# --- Correlation Map (Heatmap) ---
plt.figure(figsize=(14, 9))
sns.heatmap(data.corr(), annot=True, cmap='Reds', linewidths=0.1)
plt.suptitle('Correlation Map of Numerical Variables', fontweight='heavy',
             x=0.03, y=0.98, ha='left', fontsize='16', fontfamily='sans-serif',
             color=black_grad[0])
plt.title('Thalassemia has weak correlation with target. Whereas chest pain,␣
 ↪the maximum heart rate achieved and the slope has moderate correlation with␣
 ↪heart disease. ',
          fontsize='15', fontfamily='sans-serif', loc='left',␣
 ↪color=black_grad[1])
plt.tight_layout(rect=[0, 0.04, 1, 1.01])
```



PairPlot for quantative variables

[39]:
```
df_features = ['age', 'trestbps', 'chol', 'thalach','oldpeak', 'target']
sns.set_style('darkgrid')
sns.pairplot(data[df_features], hue = 'target', palette = 'rocket_r')
```

[39]: <seaborn.axisgrid.PairGrid at 0x7f34d4f82e50>

We will adjust the names so they can be more readable

[40]:
```
data_re = data.rename(columns = {"age":"Age", "sex":"Gender", "cp":"Chest Pain␣
↪Type",
          "trestbps":"Resting Blood Pressure", "chol":"Serum Cholesterol",␣
↪"fbs":"Fasting Blood Sugar",
          "restecg":'Resting Electrocardiographic',
          "thalach":'Max Heart Rate Achieved',
          "exang":'Exercise Induced Angina',
          "oldpeak":'ST depression induced by Exercise relative to Rest',
          "slope":'Slope of Peak Exercise ST Segment',
          "ca":'# Major Vessels colored by Flouroscopy',
          "thal":'Thalassemia (norm.,fix. defect, revers. defect)',
          "target":'Diagnosis'})
data_re.head()
```

[40]:
```
   Age  Gender  Chest Pain Type  Resting Blood Pressure  Serum Cholesterol  \
0   63       1                3                     145                233
1   37       1                2                     130                250
2   41       0                1                     130                204
3   56       1                1                     120                236
4   57       0                0                     120                354

   Fasting Blood Sugar  Resting Electrocardiographic  Max Heart Rate Achieved  \
0                    1                             0                      150
1                    0                             1                      187
2                    0                             0                      172
3                    0                             1                      178
4                    0                             1                      163

   Exercise Induced Angina  \
0                        0
1                        0
2                        0
3                        0
4                        1

   ST depression induced by Exercise relative to Rest  \
0                                                2.3
1                                                3.5
2                                                1.4
3                                                0.8
4                                                0.6

   Slope of Peak Exercise ST Segment  # Major Vessels colored by Flouroscopy  \
0                                  0                                        0
1                                  0                                        0
2                                  2                                        0
3                                  2                                        0
4                                  2                                        0

   Thalassemia (norm.,fix. defect, revers. defect)  Diagnosis
0                                                1          1
1                                                2          1
2                                                2          1
3                                                2          1
4                                                2          1
```

[41]:
```
data_re.columns
```

[41]:
```
Index(['Age', 'Gender', 'Chest Pain Type', 'Resting Blood Pressure',
       'Serum Cholesterol', 'Fasting Blood Sugar',
       'Resting Electrocardiographic', 'Max Heart Rate Achieved',
```

```
 'Exercise Induced Angina',
 'ST depression induced by Exercise relative to Rest',
 'Slope of Peak Exercise ST Segment',
 '# Major Vessels colored by Flouroscopy',
 'Thalassemia (norm.,fix. defect, revers. defect)', 'Diagnosis'],
dtype='object')
```

9.Dataset Pre-processing **This section will prepare the dataset before building the machine learning models.**

Some of the categorical values as described above have ONLY A FEW unique values. It is good practice to use Categorical Encoding for these so that the ML algorithms do not overfit to unique values. Making these into binary allows the ML algorithms to process the data in a less biased manner without losing any of the information.

thal: 'Thalassemia (norm.,fix. defect, revers. defect)'

[42]:
```
data_re['Thalassemia (norm.,fix. defect, revers. defect)'].unique()
```

[42]:
```
array([1, 2, 3, 0])
```

[43]:
```
data_re['No Thalassemia']=np.where((data_re['Thalassemia (norm.,fix. defect,␣
↪revers. defect)']==0), 1, 0)
data_re['Thalassemia: normal']=np.where((data_re['Thalassemia (norm.,fix.␣
↪defect, revers. defect)']==1), 1, 0)
data_re['Thalassemia: fixed defect']=np.where((data_re['Thalassemia (norm.,fix.␣
↪defect, revers. defect)']==2), 1, 0)
data_re['Thalassemia: reversable defect']=np.where((data_re['Thalassemia (norm.
↪,fix. defect, revers. defect)']==3), 1, 0)
data_re.drop(['Thalassemia (norm.,fix. defect, revers.␣
↪defect)'],axis=1,inplace=True)
```

ca: '# Major Vessels colored by Flouroscopy'

[44]:
```
data_re['Major Vessels F. Colored: 0']=np.where((data_re['# Major Vessels␣
↪colored by Flouroscopy']==0), 1, 0)
data_re['Major Vessel F. Colored: 1']=np.where((data_re['# Major Vessels␣
↪colored by Flouroscopy']==1), 1, 0)
data_re['Major Vessels F. Colored: 2']=np.where((data_re['# Major Vessels␣
↪colored by Flouroscopy']==2), 1, 0)
data_re['Major Vessels F. Colored: 3']=np.where((data_re['# Major Vessels␣
↪colored by Flouroscopy']==3), 1, 0)
data_re.drop(['# Major Vessels colored by Flouroscopy'],axis=1,inplace=True)
```

slope: 'Slope of Peak Exercise ST Segment'

[45]:
```
data_re['Upslopping for Peak Exercise ST Segment']=np.where((data_re['Slope of␣
↪Peak Exercise ST Segment']==0), 1, 0)
data_re['Flat for Peak Exercise ST Segment']=np.where((data_re['Slope of Peak␣
↪Exercise ST Segment']==1), 1, 0)
data_re['Downslopping for Peak Exercise ST Segment']=np.where((data_re['Slope␣
↪of Peak Exercise ST Segment']==2), 1, 0)
data_re.drop(['Slope of Peak Exercise ST Segment'],axis=1,inplace=True)
```

restecg: 'Resting Electrocardiographic'

[46]:
```
data_re['Resting Electrocardiographic: normal']=np.where((data_re['Resting␣
↪Electrocardiographic']==0), 1, 0)
data_re['Resting Electrocardiographic: ST-T wave abnormal']=np.
↪where((data_re['Resting Electrocardiographic']==1), 1, 0)
data_re['Resting Electrocardiographic: left ventricular hypertrophy']=np.
↪where((data_re['Resting Electrocardiographic']==2), 1, 0)
data_re.drop(['Resting Electrocardiographic'],axis=1,inplace=True)
```

cp: 'Chest Pain Type'

[47]:
```
data_re['Chest Pain Type: typical angina']=np.where((data_re['Chest Pain␣
↪Type']==0), 1, 0)
data_re['Chest Pain Type: atypical angina']=np.where((data_re['Chest Pain␣
↪Type']==1), 1, 0)
data_re['Chest Pain Type: non-anginal pain']=np.where((data_re['Chest Pain␣
↪Type']==2), 1, 0)
data_re['Chest Pain Type: asymptomatic']=np.where((data_re['Chest Pain␣
↪Type']==3), 1, 0)
data_re.drop(['Chest Pain Type'],axis=1,inplace=True)
```

[48]:
```
data_re.head()
```

[48]:
```
   Age  Gender  Resting Blood Pressure  Serum Cholesterol  \
0   63       1                     145                233
1   37       1                     130                250
2   41       0                     130                204
3   56       1                     120                236
4   57       0                     120                354

   Fasting Blood Sugar  Max Heart Rate Achieved  Exercise Induced Angina  \
0                    1                      150                        0
1                    0                      187                        0
2                    0                      172                        0
3                    0                      178                        0
4                    0                      163                        1
```

```
   ST depression induced by Exercise relative to Rest  Diagnosis  \
0                                                2.3          1
1                                                3.5          1
2                                                1.4          1
3                                                0.8          1
4                                                0.6          1

   No Thalassemia  …  Upslopping for Peak Exercise ST Segment  \
0               0  …                                        1
1               0  …                                        1
2               0  …                                        0
3               0  …                                        0
4               0  …                                        0

   Flat for Peak Exercise ST Segment  \
0                                  0
1                                  0
2                                  0
3                                  0
4                                  0

   Downslopping for Peak Exercise ST Segment  \
0                                          0
1                                          0
2                                          1
3                                          1
4                                          1

   Resting Electrocardiographic: normal  \
0                                     1
1                                     0
2                                     1
3                                     0
4                                     0

   Resting Electrocardiographic: ST-T wave abnormal  \
0                                                 0
1                                                 1
2                                                 0
3                                                 1
4                                                 1

   Resting Electrocardiographic: left ventricular hypertrophy  \
0                                                  0
1                                                  0
2                                                  0
3                                                  0
```

```
4                                                  0

   Chest Pain Type: typical angina  Chest Pain Type: atypical angina  \
0                                0                                 0
1                                0                                 0
2                                0                                 1
3                                0                                 1
4                                1                                 0

   Chest Pain Type: non-anginal pain  Chest Pain Type: asymptomatic
0                                  0                              1
1                                  1                              0
2                                  0                              0
3                                  0                              0
4                                  0                              0

[5 rows x 27 columns]
```

[49]:
```
data_re.columns
```

[49]:
```
Index(['Age', 'Gender', 'Resting Blood Pressure', 'Serum Cholesterol',
       'Fasting Blood Sugar', 'Max Heart Rate Achieved',
       'Exercise Induced Angina',
       'ST depression induced by Exercise relative to Rest', 'Diagnosis',
       'No Thalassemia', 'Thalassemia: normal', 'Thalassemia: fixed defect',
       'Thalassemia: reversable defect', 'Major Vessels F. Colored: 0',
       'Major Vessel F. Colored: 1', 'Major Vessels F. Colored: 2',
       'Major Vessels F. Colored: 3',
       'Upslopping for Peak Exercise ST Segment',
       'Flat for Peak Exercise ST Segment',
       'Downslopping for Peak Exercise ST Segment',
       'Resting Electrocardiographic: normal',
       'Resting Electrocardiographic: ST-T wave abnormal',
       'Resting Electrocardiographic: left ventricular hypertrophy',
       'Chest Pain Type: typical angina', 'Chest Pain Type: atypical angina',
       'Chest Pain Type: non-anginal pain', 'Chest Pain Type: asymptomatic'],
      dtype='object')
```

Splitting dataset in features and target variable

[50]:
```
feature_cols = ['Age', 'Gender', 'Resting Blood Pressure', 'Serum Cholesterol',
       'Fasting Blood Sugar', 'Max Heart Rate Achieved',
       'Exercise Induced Angina',
       'ST depression induced by Exercise relative to Rest',
       'Thalassemia: normal', 'Thalassemia: fixed defect',
       'Thalassemia: reversable defect', 'Major Vessels F. Colored: 0',
       'Major Vessel F. Colored: 1', 'Major Vessels F. Colored: 2',
```

```
        'Major Vessels F. Colored: 3',
        'Upslopping for Peak Exercise ST Segment',
        'Flat for Peak Exercise ST Segment',
        'Downslopping for Peak Exercise ST Segment',
        'Resting Electrocardiographic: normal',
        'Resting Electrocardiographic: ST-T wave abnormal',
        'Resting Electrocardiographic: left ventricular hypertrophy',
        'Chest Pain Type: typical angina', 'Chest Pain Type: atypical angina',
        'Chest Pain Type: non-anginal pain', 'Chest Pain Type: asymptomatic']

X = data_re[feature_cols].values # Features

y = data_re.Diagnosis.values # Target variable
```

Data Normalization It is important to scale the data so the ML algorithms do not overfit to the wrong features. Using the MinMaxScaler() , the values are scaled per feature based on the minimum and maximum between 0 and 1. This keeps the information from being lost but allows the ML algorithms to correctly train with the data.

[51]:
```
X = data_re.drop(['Diagnosis'], axis= 1)
y= pd.DataFrame(data_re['Diagnosis'])
```

[52]:
```
#Scale Data
scaler = MinMaxScaler()
X=MinMaxScaler().fit_transform(X.values)
X = pd.DataFrame(X)
X.columns=(data_re.drop(['Diagnosis'], axis= 1)).columns
Xy=pd.concat([y,X],axis=1)
```

[53]:
```
fix,ax = plt.subplots(figsize=(22,22))
heatmap_data = Xy
sns.heatmap(heatmap_data.corr(),vmax=1,linewidths=0.01,
            square=True,annot=True,linecolor="white", cmap='Reds')
bottom,top=ax.get_ylim()
ax.set_ylim(bottom+0.5,top-0.5)
heatmap_title='Figure 1:  Heatmap of Pearson Correlation Coefficient Matrix for␣
 ↪Features'
ax.set_title(heatmap_title)
plt.show()
```

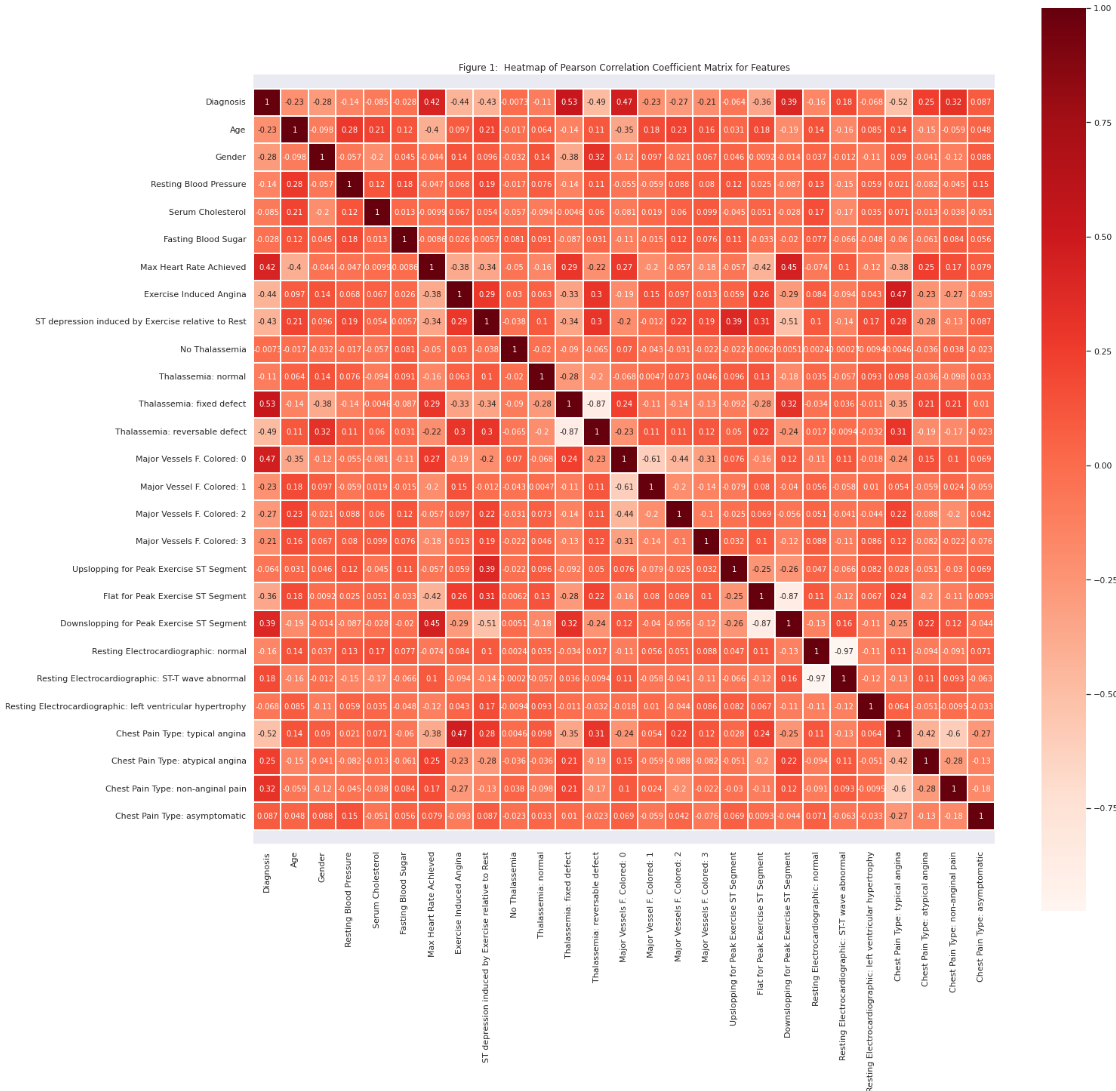


Figure 1: Heatmap of Pearson Correlation Coefficient Matrix for Features

After encoding features, we can have a better insight for the correlation between variables and the diagnosis. It shows here that 'Thalassemia :fixed defect' and 'Major vessels F. colored:0' have the strongest corroletion with 'Diagnosis'. Chest Pain Type: typical angina has the lowest one.

Splitting the Dataset The Data was split into 80% training (237 people) and 20% testing (60 people) after dropping 6 instances with missing values. This is a general rule of thumb for splitting data to train ML algorithms with.

[54]:
```
# split X and y into training and testing sets
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)
```

[55]:
```
X_train.shape
```

[55]: (242, 26)

[56]:
```
X_test.shape
```

[56]: (61, 26)

[57]:
```
y_train.shape
```

[57]: (242, 1)

[58]:
```
y_test.shape
```

[58]: (61, 1)

Model Implementation **Logistic Regression** Logistic regression is a statistical method that is used for building machine learning models where the dependent variable is dichotomous: i.e. binary. Logistic regression is used to describe data and the relationship between one dependent variable and one or more independent variables. The independent variables can be nominal, ordinal, or of interval type.

The name "logistic regression" is derived from the concept of the logistic function that it uses. The logistic function is also known as the sigmoid function. The value of this logistic function lies between zero and one.

[59]:
```
# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)
```

[60]:
```
y_pred=logreg.predict(X_test)
y_pred
```

[60]:
```
array([1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0])
```

[61]:
```
y_preda=logreg.predict_proba(X_test)
y_preda
```

```
[61]: array([[0.17419153, 0.82580847],
       [0.77068705, 0.22931295],
       [0.00645951, 0.99354049],
       [0.58274664, 0.41725336],
       [0.29029426, 0.70970574],
       [0.96629939, 0.03370061],
       [0.58165332, 0.41834668],
       [0.53569786, 0.46430214],
       [0.05246778, 0.94753222],
       [0.04024922, 0.95975078],
       [0.0205683 , 0.9794317 ],
       [0.02594463, 0.97405537],
       [0.25430101, 0.74569899],
       [0.43910985, 0.56089015],
       [0.9348479 , 0.0651521 ],
       [0.3686415 , 0.6313585 ],
       [0.21093822, 0.78906178],
       [0.39753885, 0.60246115],
       [0.24031828, 0.75968172],
       [0.6266869 , 0.3733131 ],
       [0.13018072, 0.86981928],
       [0.98296919, 0.01703081],
       [0.14197329, 0.85802671],
       [0.80120337, 0.19879663],
       [0.03041825, 0.96958175],
       [0.26702983, 0.73297017],
       [0.55101294, 0.44898706],
       [0.78379948, 0.21620052],
       [0.89828047, 0.10171953],
       [0.08337948, 0.91662052],
       [0.96823248, 0.03176752],
       [0.05626184, 0.94373816],
       [0.10099904, 0.89900096],
       [0.01444017, 0.98555983],
       [0.02270375, 0.97729625],
       [0.97173717, 0.02826283],
       [0.93790887, 0.06209113],
       [0.09343863, 0.90656137],
       [0.96221026, 0.03778974],
       [0.98987115, 0.01012885],
       [0.27550893, 0.72449107],
       [0.28445747, 0.71554253],
       [0.03823209, 0.96176791],
       [0.58478897, 0.41521103],
       [0.01443042, 0.98556958],
       [0.52122526, 0.47877474],
       [0.00820817, 0.99179183],
```

```
        [0.0082717 , 0.9917283 ],
        [0.92137037, 0.07862963],
        [0.70970348, 0.29029652],
        [0.17824504, 0.82175496],
        [0.944841  , 0.055159  ],
        [0.04471043, 0.95528957],
        [0.01062834, 0.98937166],
        [0.9837039 , 0.0162961 ],
        [0.16406884, 0.83593116],
        [0.05054383, 0.94945617],
        [0.9611136 , 0.0388864 ],
        [0.54221226, 0.45778774],
        [0.55271441, 0.44728559],
        [0.95029225, 0.04970775]])
```

[62]:

```
# Import the necessary modules
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
cnf = confusion_matrix(y_test, y_pred)
print(confusion_matrix(y_test, y_pred))

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

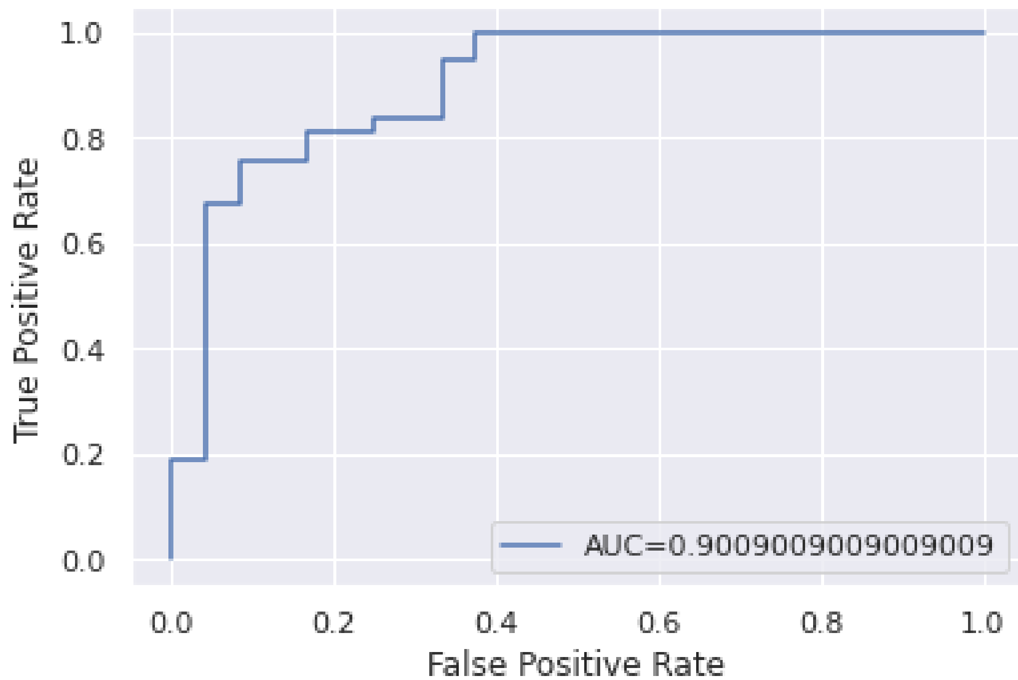
```
[[20  4]
 [ 7 30]]
              precision    recall  f1-score   support

           0       0.74      0.83      0.78        24
           1       0.88      0.81      0.85        37

    accuracy                           0.82        61
   macro avg       0.81      0.82      0.81        61
weighted avg       0.83      0.82      0.82        61
```

Logistic regression accuracy: 82%

[63]:

```
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt

from sklearn.metrics import roc_auc_score

#define metrics
y_pred_proba = logreg.predict_proba(X_test)[:,1]
fpr, tpr, _ = roc_curve(y_test,  y_pred_proba)
auc = roc_auc_score(y_test, y_pred_proba)
#area under the curve
```

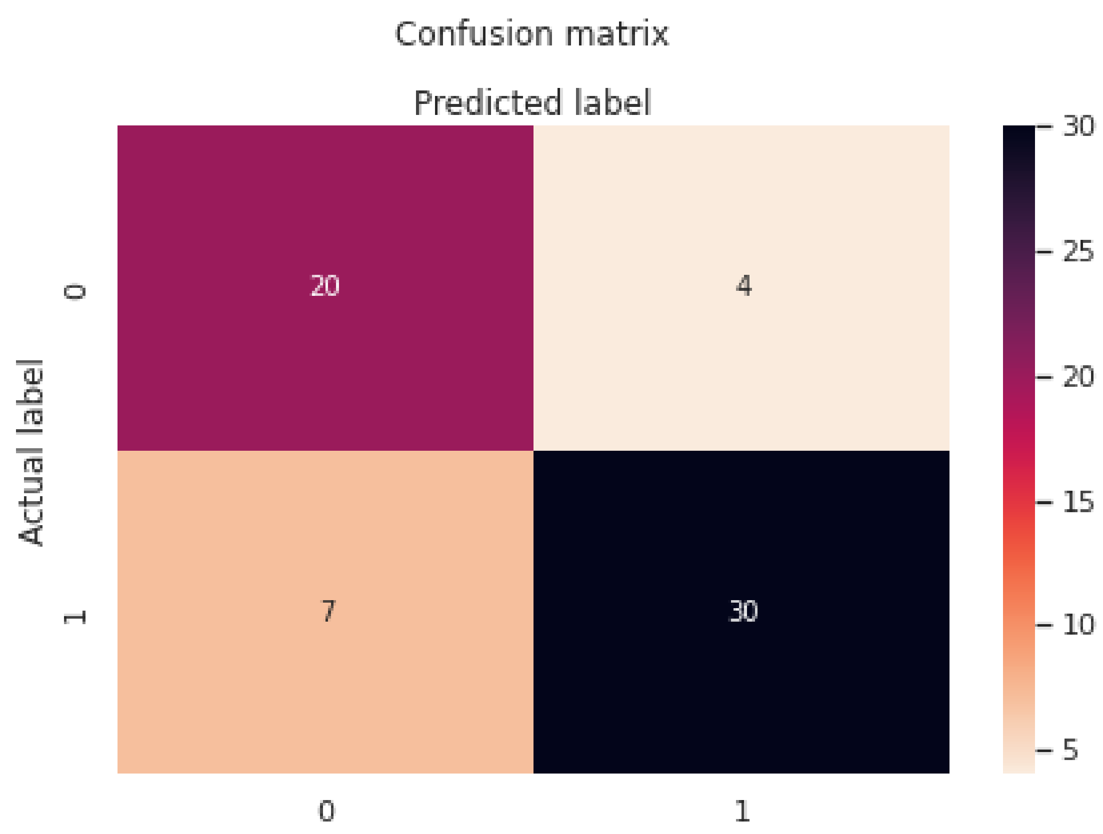
```
#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))

plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)

plt.show()
```



Area Under Curve is 0.9 which is great

[64]:
```
# cm = confusion_matrix(y_valid, y_pred)
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
sns.heatmap(cnf, annot = True,cmap="rocket_r" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

[64]:
```
Text(0.5, 257.44, 'Predicted label')
```



[65]:
```
import statsmodels.api as sm
res = sm.Logit(y,X).fit()
res.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.296420
         Iterations 17
```

[65]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              Diagnosis   No. Observations:                  303
Model:                          Logit   Df Residuals:                      280
Method:                           MLE   Df Model:                           22
Date:                Sun, 04 Sep 2022   Pseudo R-squ.:                  0.5699
Time:                        17:36:23   Log-Likelihood:                -89.815
converged:                       True   LL-Null:                       -208.82
Covariance Type:            nonrobust   LLR p-value:                 3.556e-38
================================================================================
==============================================
```

```
                                                                    coef    std err
z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------
----------------------------------------------
Age                                                               1.3353      1.221
1.094      0.274      -1.057       3.727
Gender                                                           -1.8623      0.571
-3.262      0.001      -2.981      -0.743
Resting Blood Pressure                                           -2.7732      1.266
-2.191      0.028      -5.254      -0.292
Serum Cholesterol                                                -1.8794      1.859
-1.011      0.312      -5.523       1.764
Fasting Blood Sugar                                               0.4457      0.588
0.758      0.448      -0.707       1.598
Max Heart Rate Achieved                                           2.6272      1.554
1.691      0.091      -0.418       5.672
Exercise Induced Angina                                          -0.7791      0.452
-1.724      0.085      -1.665       0.106
ST depression induced by Exercise relative to Rest               -2.4625      1.503
-1.639      0.101      -5.407       0.482
No Thalassemia                                                   -2.6861   9.04e+06
-2.97e-07      1.000   -1.77e+07    1.77e+07
Thalassemia: normal                                              -0.0485    8.1e+06
-5.99e-09      1.000   -1.59e+07    1.59e+07
Thalassemia: fixed defect                                        -0.3183   8.65e+06
-3.68e-08      1.000    -1.7e+07     1.7e+07
Thalassemia: reversable defect                                   -1.7710   8.44e+06
-2.1e-07      1.000   -1.66e+07    1.66e+07
Major Vessels F. Colored: 0                                      -1.2680      1.720
-0.737      0.461      -4.639       2.103
Major Vessel F. Colored: 1                                       -3.6103      1.785
-2.022      0.043      -7.109      -0.112
Major Vessels F. Colored: 2                                      -4.7511      1.941
-2.448      0.014      -8.555      -0.947
Major Vessels F. Colored: 3                                      -3.5151      1.926
-1.825      0.068      -7.290       0.260
Upslopping for Peak Exercise ST Segment                           2.9174   1.11e+07
2.62e-07      1.000   -2.18e+07    2.18e+07
Flat for Peak Exercise ST Segment                                 2.1424   1.09e+07
1.96e-07      1.000   -2.14e+07    2.14e+07
Downslopping for Peak Exercise ST Segment                         3.6074   1.08e+07
3.33e-07      1.000   -2.12e+07    2.12e+07
Resting Electrocardiographic: normal                              1.0463   1.02e+07
1.03e-07      1.000   -1.99e+07    1.99e+07
Resting Electrocardiographic: ST-T wave abnormal                  1.5069   9.68e+06
1.56e-07      1.000    -1.9e+07     1.9e+07
Resting Electrocardiographic: left ventricular hypertrophy        0.3321   9.66e+06
```

```
3.44e-08      1.000   -1.89e+07    1.89e+07
Chest Pain Type: typical angina                              -0.5999        nan
nan         nan          nan         nan
Chest Pain Type: atypical angina                              0.2648        nan
nan         nan          nan         nan
Chest Pain Type: non-anginal pain                             1.4033        nan
nan         nan          nan         nan
Chest Pain Type: asymptomatic                                 1.8172        nan
nan         nan          nan         nan
================================================================================
==============================================
"""
```

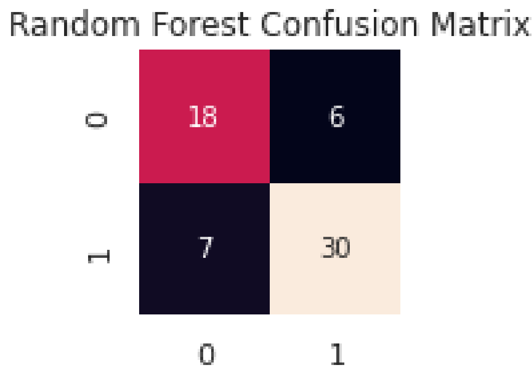
Random Forest Random Forest is a tree-based machine learning algorithm that leverages the power of multiple decision trees for making decisions. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

[66]:
```
# --- Applying Random Forest ---
from sklearn.ensemble import RandomForestClassifier
RFclassifier = RandomForestClassifier(n_estimators=1000, random_state=1,␣
 ↪max_leaf_nodes=20, min_samples_split=15)

RFclassifier.fit(X_train, y_train)
y_pred_RF = RFclassifier.predict(X_test)
```

[67]:
```
# --- Random Forest Accuracy ---
RFAcc = accuracy_score(y_pred_RF, y_test)
print('.:. Random Forest Accuracy:'+'\033[1m {:.2f}%'.format(RFAcc*100)+' .:.')
```

```
.:. Random Forest Accuracy: 78.69% .:.
```

[68]:
```
from sklearn.metrics import confusion_matrix
cm_rf = confusion_matrix(y_test,y_pred_RF)
plt.subplot(2,3,6)
plt.title("Random Forest Confusion Matrix")
sns.heatmap(cm_rf,annot=True,cmap="rocket",fmt="d",cbar=False,␣
 ↪annot_kws={"size": 10})

plt.show()
```



Logistic Regression has greater accuracy than random forest.