Heart Disease Prediction

Import Libraries

In [1]:

```
# Import needed libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import re

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingCl
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, make_s
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
import sklearn.linear_model as lmm
#ensembling
from mlxtend.classifier import StackingCVClassifier

#EDA
from collections import Counter
import pandas_profiling as pp

from plotly.offline import iplot
import plotly as py
import plotly.tools as tls

import pickle
```

1. Data Preparation and Data Exploration

In [2]:

```
# Read data in the excel file
df = pd.read_csv('data.csv')
df.head()
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	2
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	1
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	2
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	1
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	1

In [3]:

```
df.shape
```

Out[3]:

```
(270, 14)
```

In [4]:

```
df.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   cp        270 non-null    int64  
 3   trestbps  270 non-null    int64  
 4   chol      270 non-null    int64  
 5   fbs       270 non-null    int64  
 6   restecg   270 non-null    int64  
 7   thalach   270 non-null    int64  
 8   exang     270 non-null    int64  
 9   oldpeak   270 non-null    float64
 10  slope     270 non-null    int64  
 11  ca        270 non-null    int64  
 12  thal      270 non-null    int64  
 13  target    270 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 29.7 KB
```

In [5]:

```
df.describe()
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	
count	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	27
mean	54.433333	0.677778	3.174074	131.344444	249.659259	0.148148	1.022222	14
std	9.109067	0.468195	0.950090	17.861608	51.686237	0.355906	0.997891	2
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	7
25%	48.000000	0.000000	3.000000	120.000000	213.000000	0.000000	0.000000	13
50%	55.000000	1.000000	3.000000	130.000000	245.000000	0.000000	2.000000	15
75%	61.000000	1.000000	4.000000	140.000000	280.000000	0.000000	2.000000	16
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	20

In [6]:

```
#Check null values
df.isnull().sum()
```

Out[6]:

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [7]:

```
df['target'].value_counts()
```

Out[7]:

```
1    150
2    120
Name: target, dtype: int64
```

In [8]:

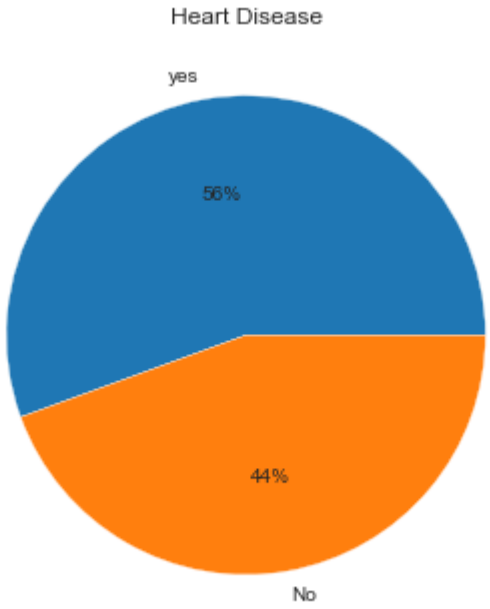
```
# Percentage of patients have and do not have heart disease
df['target'].value_counts()/df.shape[0]*100
```

Out[8]:

```
1    55.555556
2    44.444444
Name: target, dtype: float64
```

In [9]:

```
# Create a plot to display the percentage of the positive and negative heart disease
labels = ['yes', 'No']
values = df['target'].value_counts().values

plt.pie(values, labels=labels, autopct='%1.0f%%')
plt.title('Heart Disease')
plt.show()
```

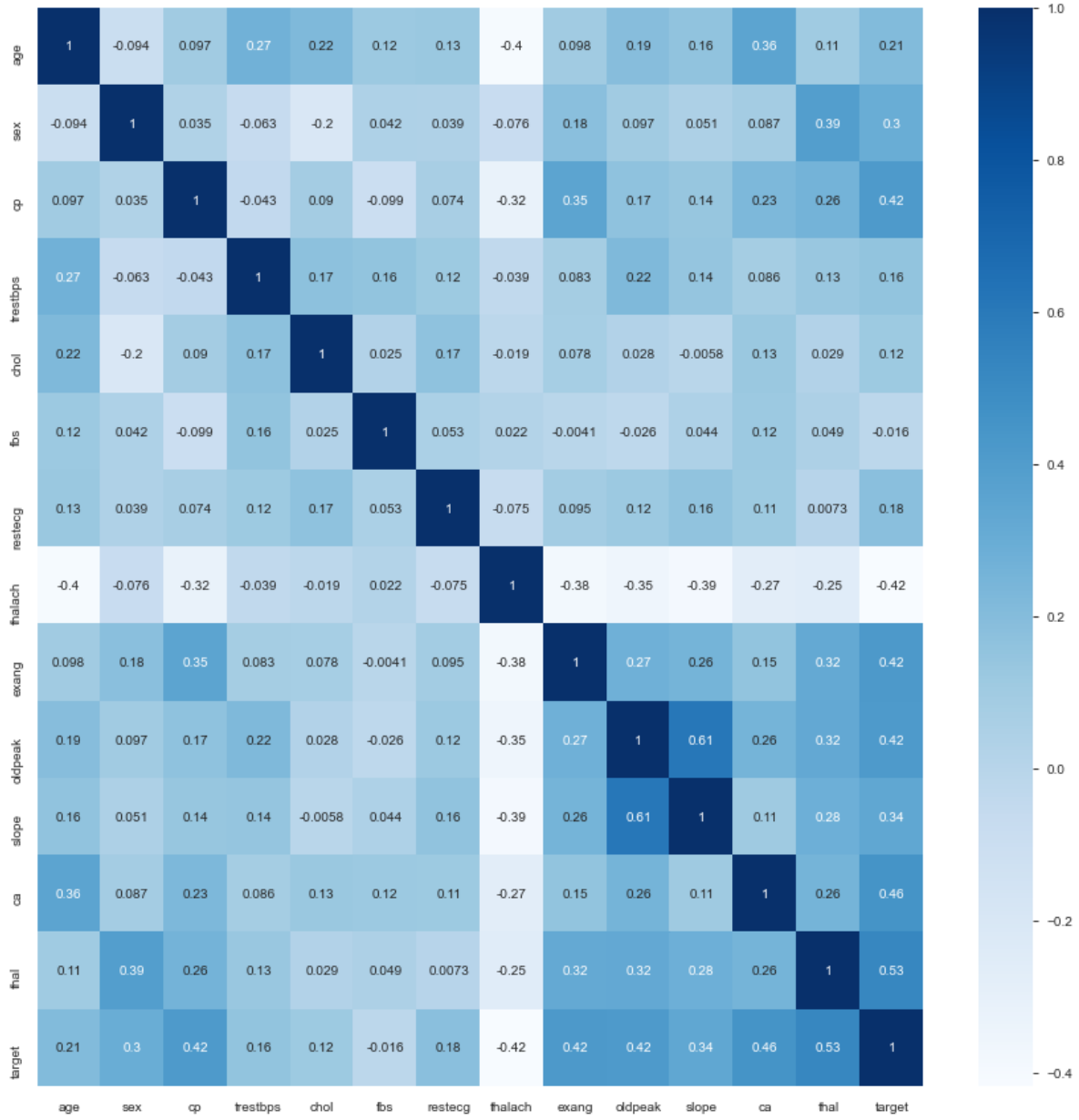


As we can see, the dataset contains 14 columns 5 of them are numerical values and 9 of them are categorical values. We can see also there are no missing values in this dataset. As for the data balancing, the data is relatively balanced, 56% of the persons in the dataset have heart disease.

Attributes Correlation

In [10]:

```
# Correlation map
plt.figure(figsize=(15, 15))
sns.heatmap(df.corr(), annot = True, cmap = "Blues")
plt.show()
```



From the above correlation plot, the chest pain type (cp), exercise induced angina (exang), ST depression induced by exercise relative to rest (oldpeak), the slope of the peak exercise ST segment (slope), number of major vessels (0-3) colored by flourosopy (ca) and thalassemia (thal) are correlated with the heart disease (target) directly. We see also that there is an inverse proportion between the heart disease and maximum heart rate (thalch).

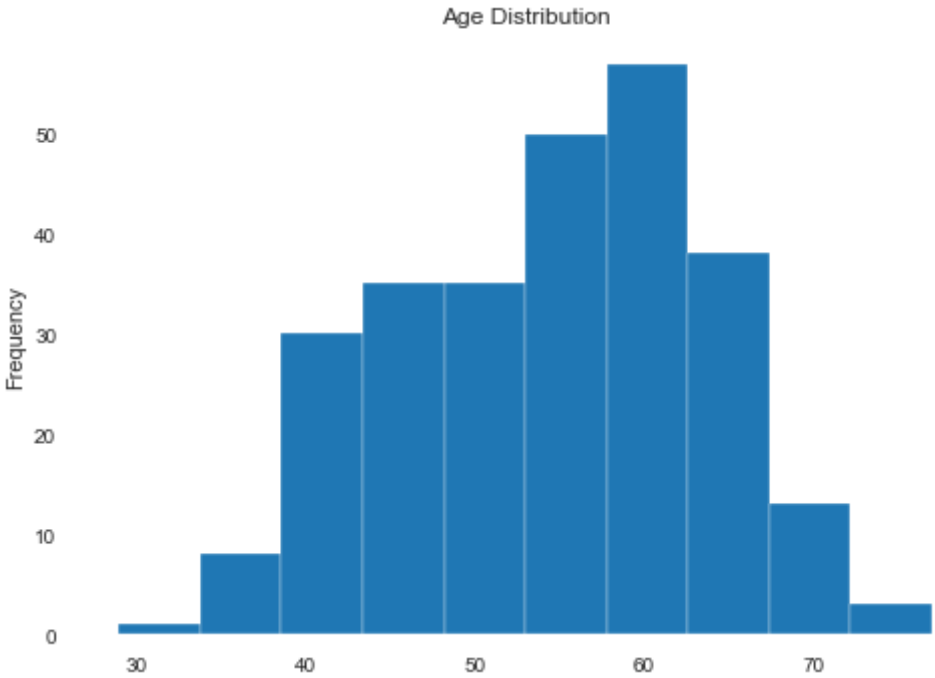
We can see also, there are a relation between the following attributes:

- The number of major vessels (0-3) colored by flourosopy (ca) and the age.
- ST depression induced by exercise relative to rest (oldpeak) and the slope of the peak exercise ST segment (slope).
- The chest pain type (cp), exercise induced angina (exang).
- maximum heart rate (thalch) and the age.

Age Exploration

In [11]:

```
# Display age distribution
df['age'].plot(kind = 'hist', title = 'Age Distribution')
plt.show()
```



In [12]:

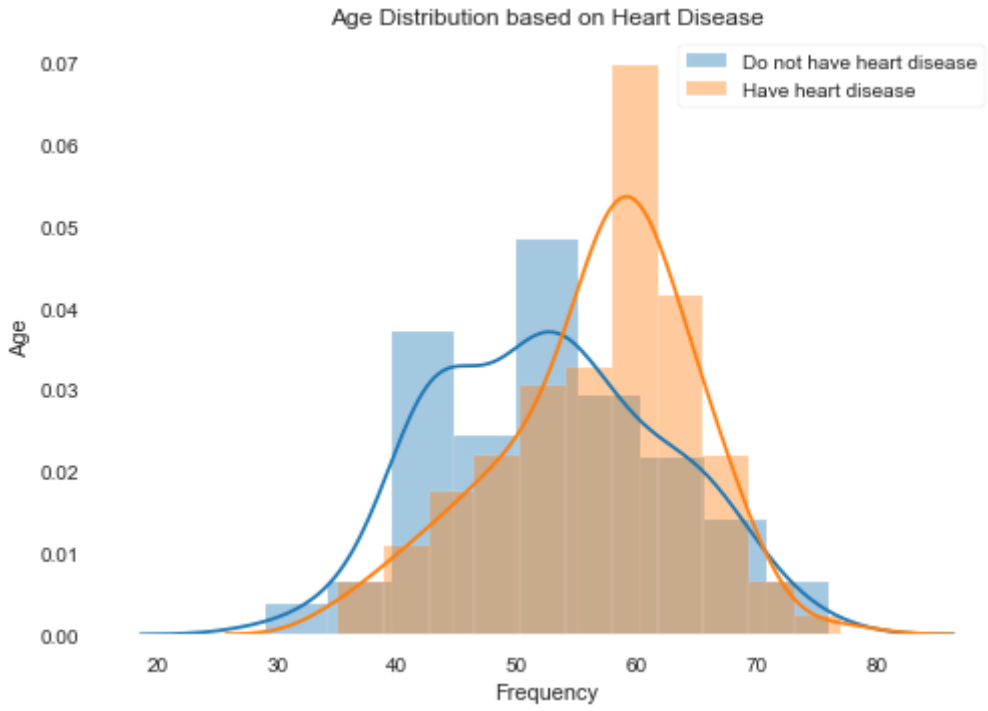
```
# Get min, max and average of the age
print('Min age: ', min(df['age']))
print('Max age: ', max(df['age']))
print('Average age: ', df['age'].mean())
```

```
Min age:  29
Max age:  77
Average age:  54.43333333333333
```

In [13]:

```
# Display age distribution based on heart disease
sns.distplot(df[df['target'] == 1]['age'], label='Do not have heart disease')
sns.distplot(df[df['target'] == 2]['age'], label = 'Have heart disease')
plt.xlabel('Frequency')
plt.ylabel('Age')
plt.title('Age Distribution based on Heart Disease')
plt.legend()
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```



In [14]:

```
# Get min, max and average of the age of the people do not have heart diseas
print('Min age of people who do not have heart disease: ', min(df[df['target'] == 1]['age']
print('Max age of people who do not have heart disease: ', max(df[df['target'] == 1]['age']
print('Average age of people who do not have heart disease: ', df[df['target'] == 1]['age']
```

```
Min age of people who do not have heart disease:  29
Max age of people who do not have heart disease:  76
Average age of people who do not have heart disease:  52.70666666666666
```

In [15]:

```
# Get min, max and average of the age of the people have heart diseas
print('Min age of people who have heart disease: ', min(df[df['target'] == 2]['age']))
print('Max age of people who have heart disease: ', max(df[df['target'] == 2]['age']))
print('Average age of people who have heart disease: ', df[df['target'] == 2]['age'].mean()
```

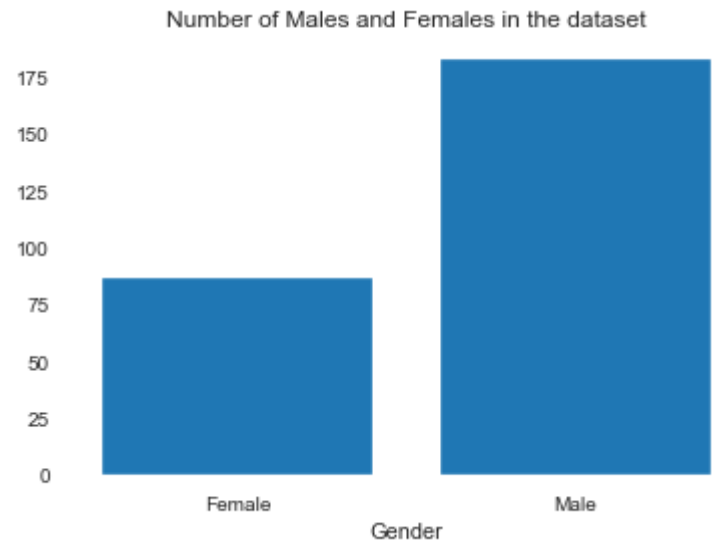
```
Min age of people who have heart disease:  35
Max age of people who have heart disease:  77
Average age of people who have heart disease:  56.59166666666667
```

From the data, I can say that the heart disease infects the old and young people, and the probability of the old people te be infected is higher than young people.

Gender Exploration

In [16]:

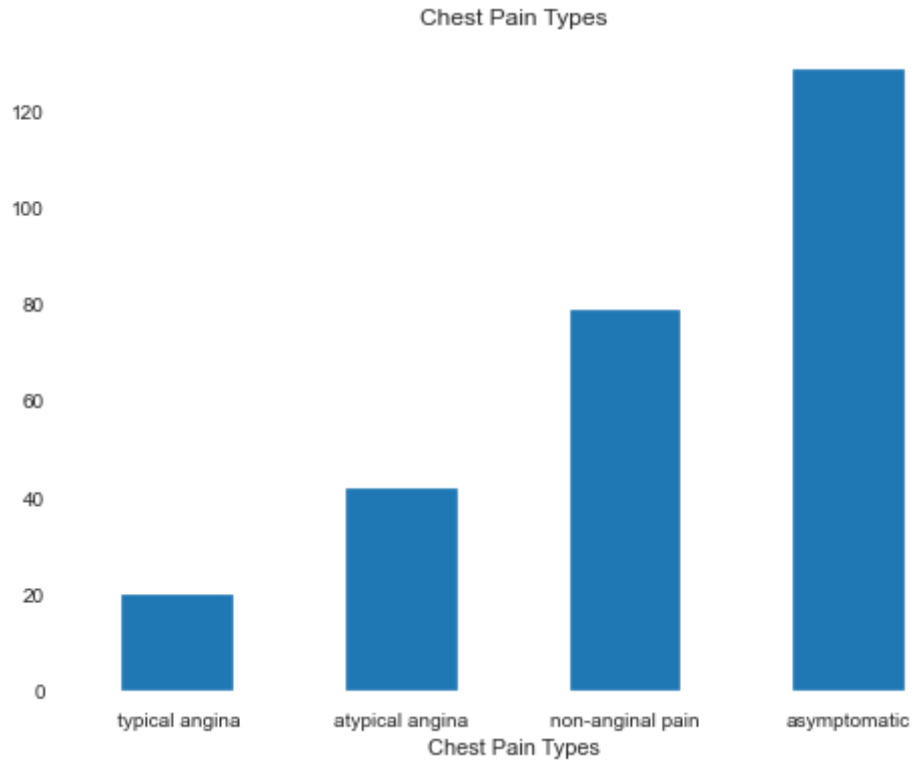
```
# Number of males and females
F = df[df['sex'] == 0].count()['target']
M = df[df['sex'] == 1].count()['target']

# Create a plot
figure, ax = plt.subplots(figsize = (6, 4))
ax.bar(x = ['Female', 'Male'], height = [F, M])
plt.xlabel('Gender')
plt.title('Number of Males and Females in the dataset')
plt.show()
```



Chest Pain Type Exploration

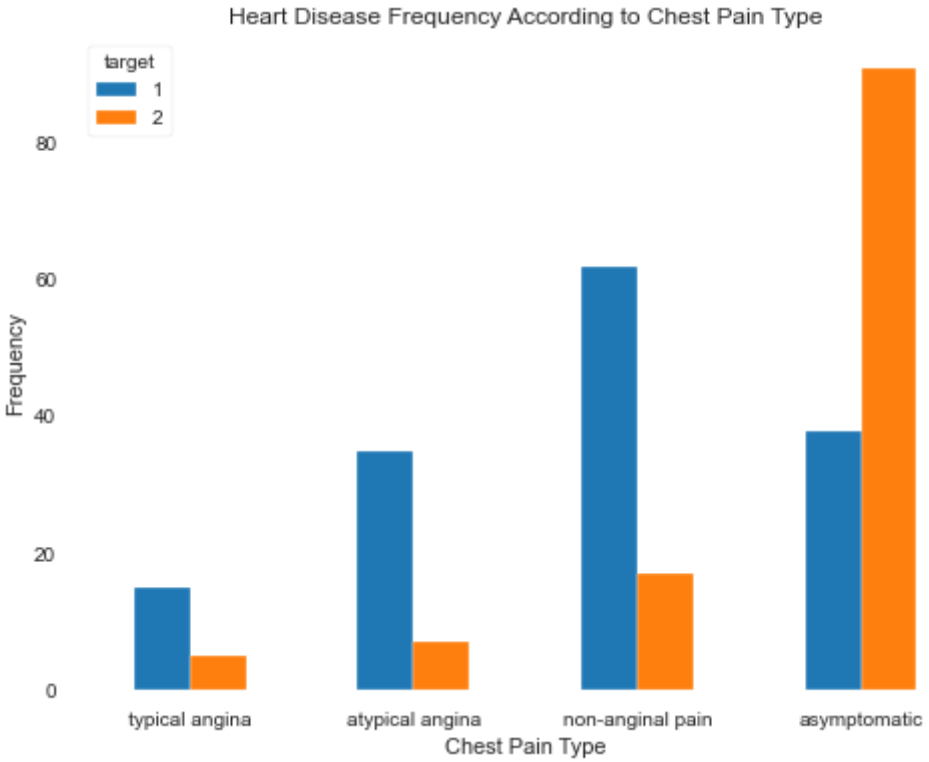
In [17]:

```
# Display chest pain types in bar chart
df.groupby(df['cp']).count()['target'].plot(kind = 'bar', title = 'Chest Pain Types', figsi
plt.xlabel('Chest Pain Types')
plt.xticks(np.arange(4), ('typical angina', 'atypical angina', 'non-anginal pain', 'asympto
plt.show()
```



In [18]:

```
# Display chest pain types based on the target
pd.crosstab(df.cp,df.target).plot(kind = "bar", figsize = (8, 6))
plt.title('Heart Disease Frequency According to Chest Pain Type')
plt.xlabel('Chest Pain Type')
plt.xticks(np.arange(4), ('typical angina', 'atypical angina', 'non-anginal pain', 'asympto
plt.ylabel('Frequency')
plt.show()
```

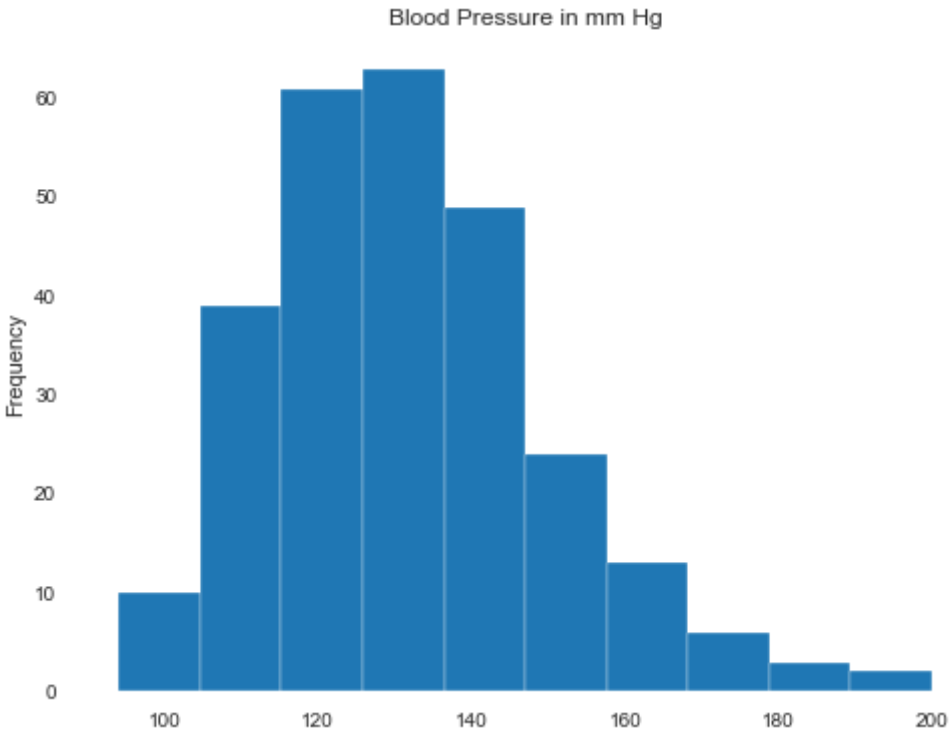


We can see that most of the people with heart disease have asymptomatic chest pain.

Blood Pressure Exploration

In [19]:

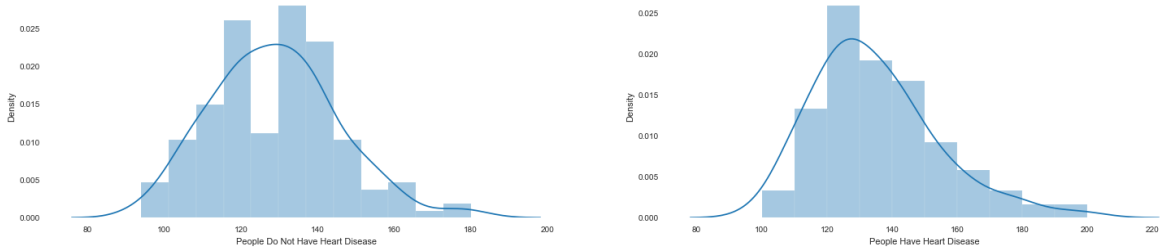
```
# Display blood pressure distribution
df['trestbps'].plot(kind = 'hist', title = 'Blood Pressure in mm Hg', figsize = (8, 6))
plt.show()
```



In [20]:

```
# Display blood pressure distribution based on heart disease
fig, (axis1, axis2) = plt.subplots(1, 2,figsize=(25, 5))
ax = sns.distplot(df[df['target'] == 1]['trestbps'], label='Do not have heart disease', ax
ax.set(xlabel='People Do Not Have Heart Disease')
ax = sns.distplot(df[df['target'] == 2]['trestbps'], label = 'Have heart disease', ax = axi
ax.set(xlabel='People Have Heart Disease')
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```



In [21]:

```
# Get min, max and average of the  blood pressure of the people do not have heart diseas
print('Min blood pressure of people who do not have heart disease: ', min(df[df['target'] =
print('Max blood pressure of people who do not have heart disease: ', max(df[df['target'] =
print('Average blood pressure of people who do not have heart disease: ', df[df['target'] =
```

```
Min blood pressure of people who do not have heart disease:  94
Max blood pressure of people who do not have heart disease:  180
Average blood pressure of people who do not have heart disease:  128.8666666
6666667
```

In [22]:

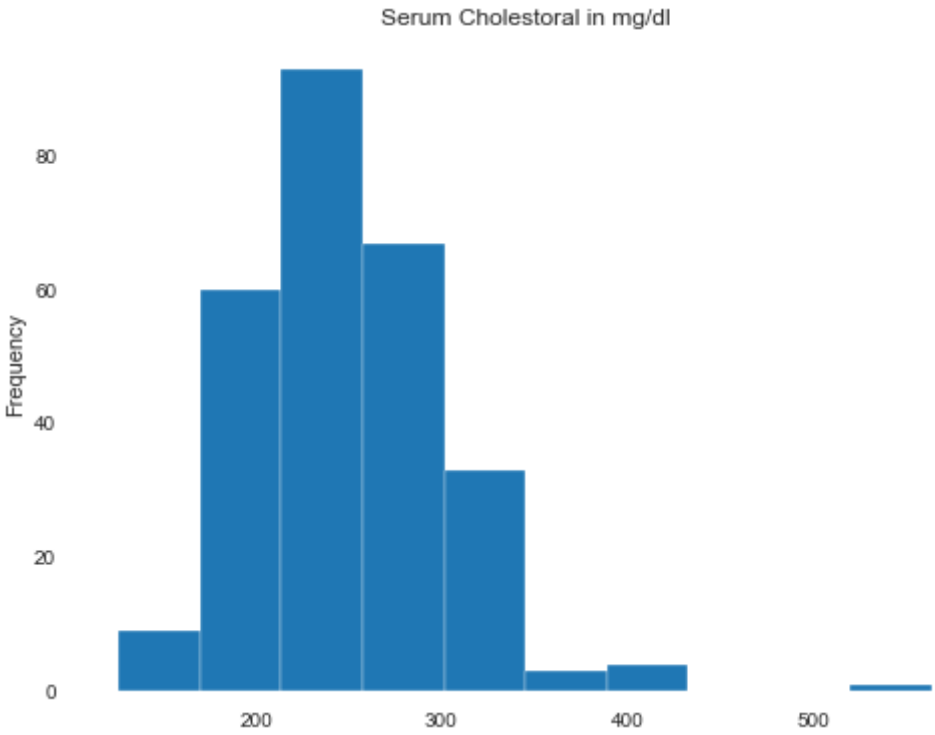
```
# Get min, max and average of the blood pressure of the people have heart diseas
print('Min blood pressure of people who have heart disease: ', min(df[df['target'] == 2]['t
print('Max blood pressure of people who have heart disease: ', max(df[df['target'] == 2]['t
print('Average blood pressure of people who have heart disease: ', df[df['target'] == 2]['t
```

```
Min blood pressure of people who have heart disease:  100
Max blood pressure of people who have heart disease:  200
Average blood pressure of people who have heart disease:  134.44166666666666
```

Cholestoral Exploration

In [23]:

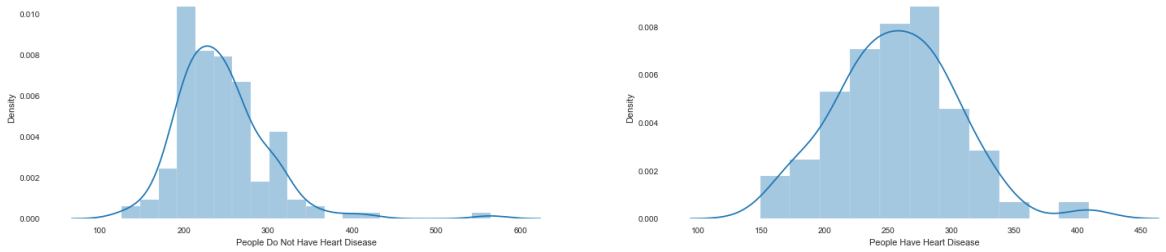
```
# Display Cholestoral distribution
df['chol'].plot(kind = 'hist', title = 'Serum Cholestoral in mg/dl', figsize = (8, 6))
plt.show()
```



In [24]:

```
# Display Cholestoral distribution based on heart disease
fig, (axis1, axis2) = plt.subplots(1, 2, figsize=(25, 5))
ax = sns.distplot(df[df['target'] == 1]['chol'], label='Do not have heart disease', ax = ax
ax.set(xlabel='People Do Not Have Heart Disease')
ax = sns.distplot(df[df['target'] == 2]['chol'], label = 'Have heart disease', ax = axis2)
ax.set(xlabel='People Have Heart Disease')
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```



In [25]:

```
# Get min, max and average of the Cholestoral of the people do not have heart diseas
print('Min cholestoral of people who do not have heart disease: ', min(df[df['target'] == 1
print('Max cholestoral of people who do not have heart disease: ', max(df[df['target'] == 1
print('Average cholestoral of people who do not have heart disease: ', df[df['target'] == 1
```

```
Min cholestoral of people who do not have heart disease:  126
Max cholestoral of people who do not have heart disease:  564
Average cholestoral of people who do not have heart disease:  244.2133333333
3334
```

In [26]:

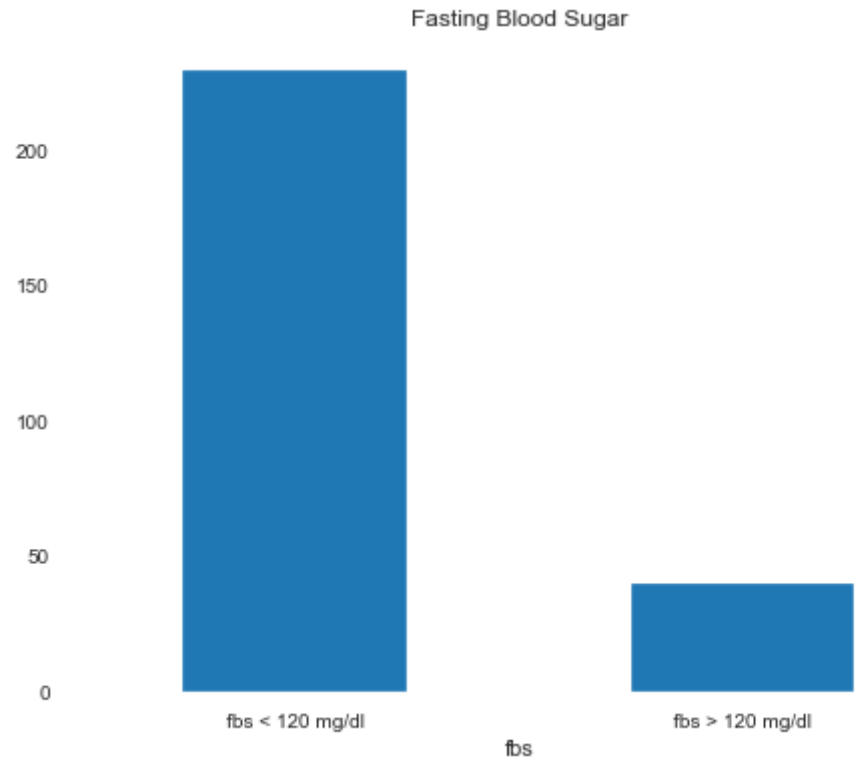
```
# Get min, max and average of the Cholestoral of the people have heart diseas
print('Min cholestoral of people who have heart disease: ', min(df[df['target'] == 2]['chol
print('Max cholestoral of people who have heart disease: ', max(df[df['target'] == 2]['chol
print('Average cholestorale of people who have heart disease: ', df[df['target'] == 2]['cho
```

```
Min cholestoral of people who have heart disease:  149
Max cholestoral of people who have heart disease:  409
Average cholestorale of people who have heart disease:  256.46666666666664
```

Fasting Blood Sugar Exploration

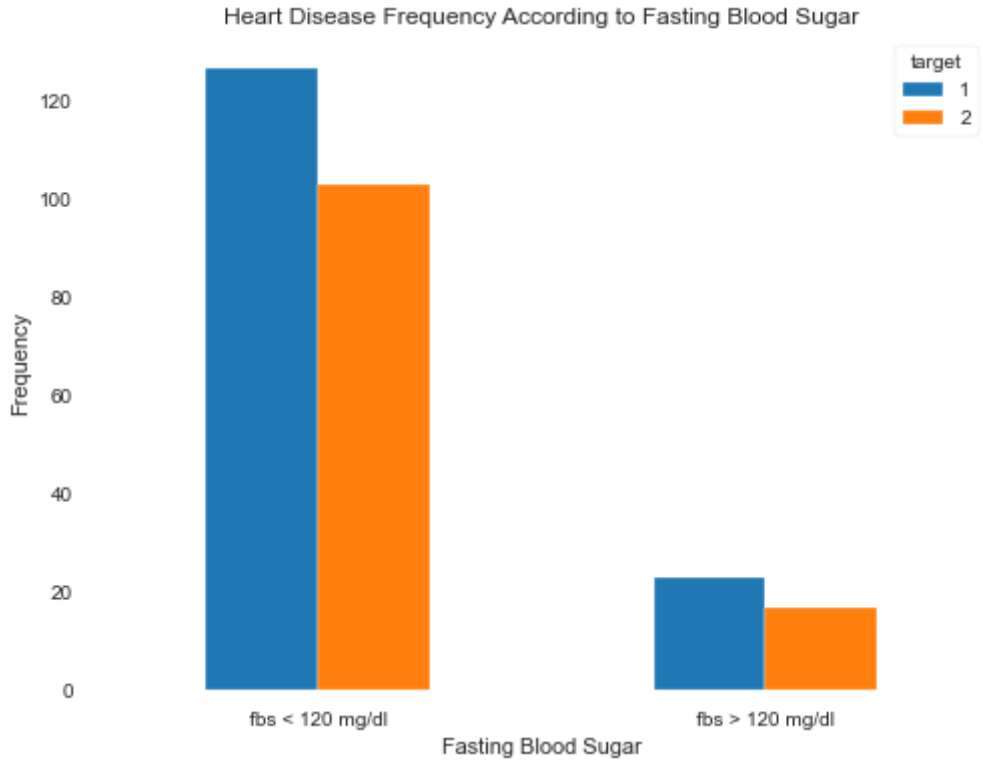
In [27]:

```
# Display fasting blood sugar in bar chart
df.groupby(df['fbs']).count()['target'].plot(kind = 'bar', title = 'Fasting Blood Sugar', f
plt.xticks(np.arange(2), ('fbs < 120 mg/dl', 'fbs > 120 mg/dl'), rotation = 0)
plt.show()
```



In [28]:

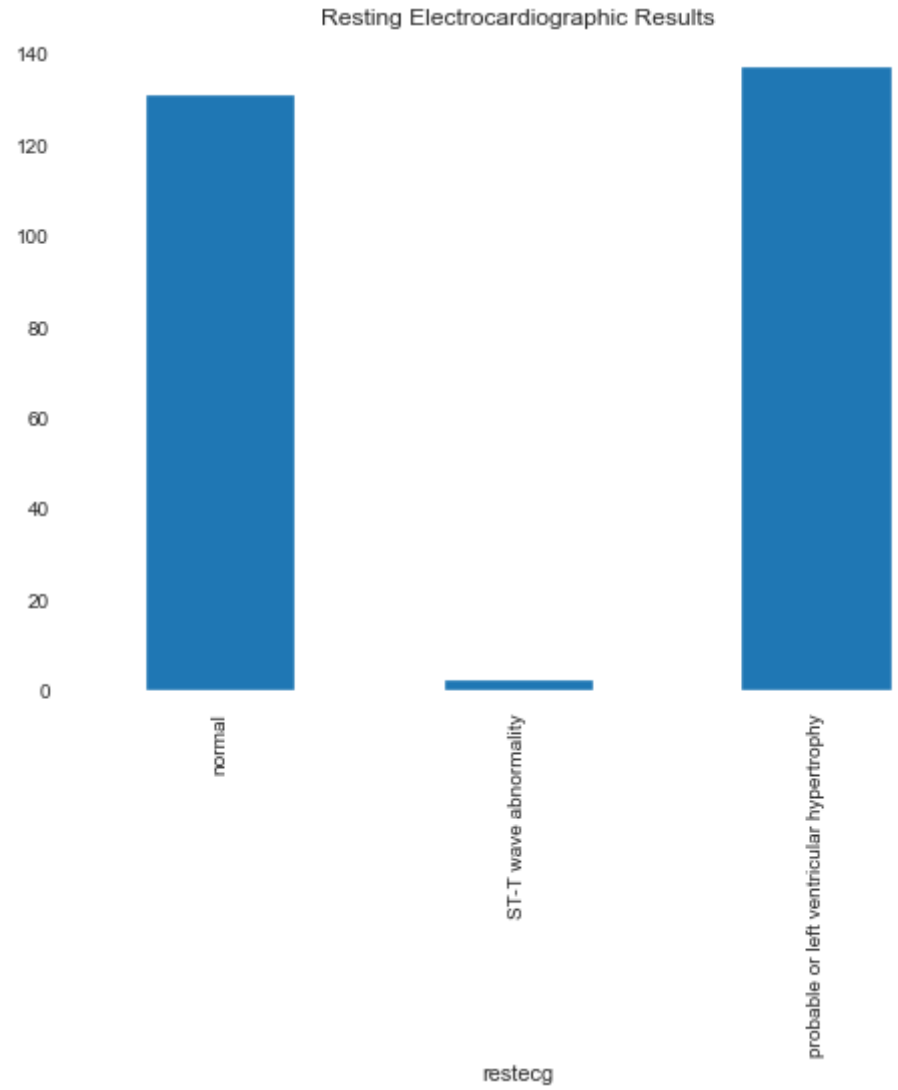
```
# Display fasting blood sugar based on the target
pd.crosstab(df.fbs,df.target).plot(kind = "bar", figsize = (8, 6))
plt.title('Heart Disease Frequency According to Fasting Blood Sugar')
plt.xlabel('Fasting Blood Sugar')
plt.xticks(np.arange(2), ('fbs < 120 mg/dl', 'fbs > 120 mg/dl'), rotation = 0)
plt.ylabel('Frequency')
plt.show()
```



Electrocardiographic Results Exploration

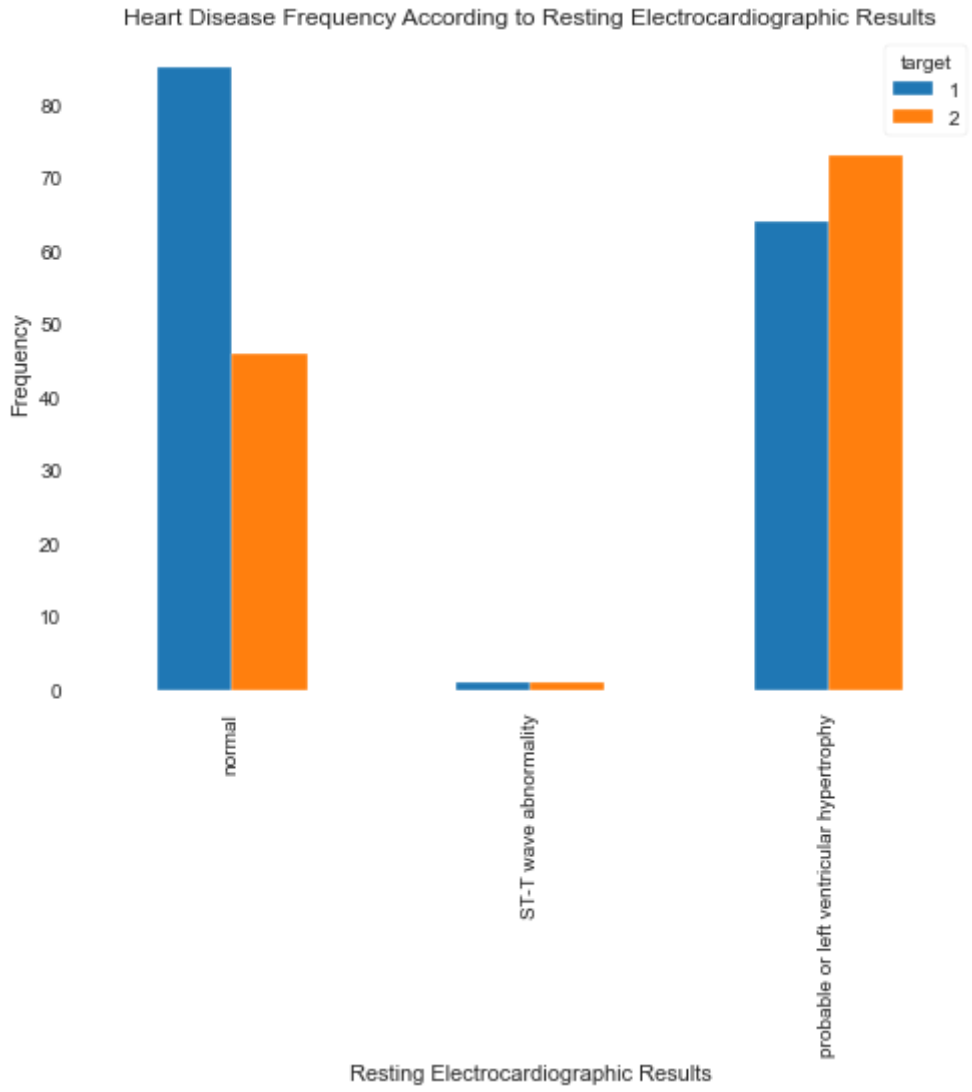
In [29]:

```
# Display electrocardiographic results in bar chart
df.groupby(df['restecg']).count()['target'].plot(kind = 'bar', title = 'Resting Electrocard
plt.xticks(np.arange(3), ('normal', 'ST-T wave abnormality', 'probable or left ventricular
plt.show()
```



In [30]:

```
# Display resting electrocardiographic results based on the target
pd.crosstab(df.restecg,df.target).plot(kind = "bar", figsize = (8, 6))
plt.title('Heart Disease Frequency According to Resting Electrocardiographic Results')
plt.xticks(np.arange(3), ('normal', 'ST-T wave abnormality', 'probable or left ventricular
plt.xlabel('Resting Electrocardiographic Results')
plt.ylabel('Frequency')
plt.show()
```

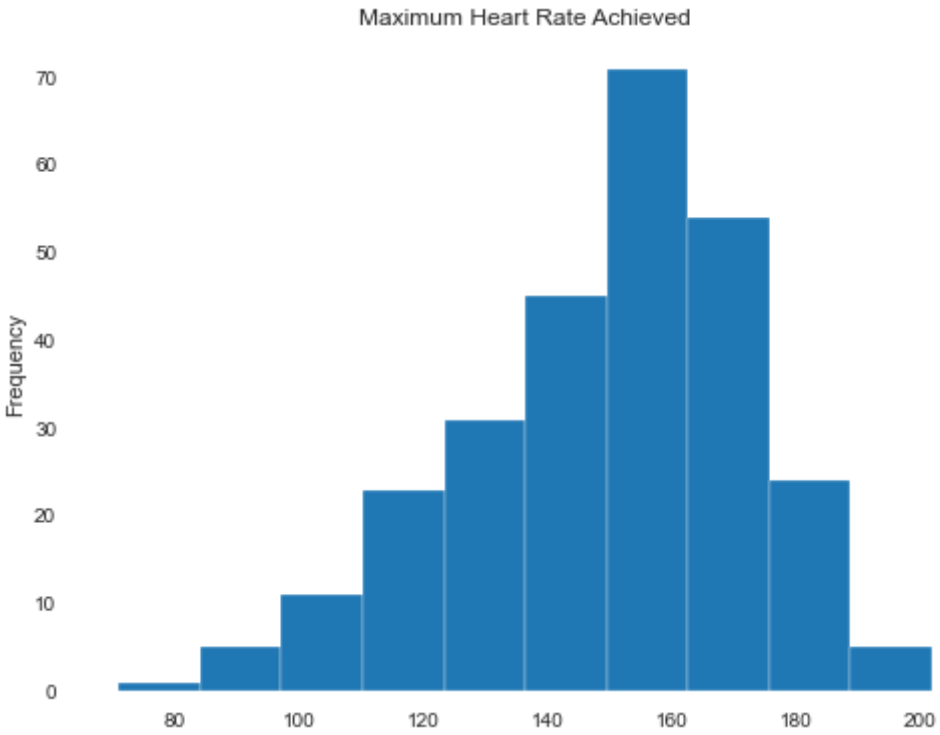


Usually the people who do not have heart disease have normal electrocardiographic, whereas the people who have heart disease have probable or left ventricular hypertrophy.

Maximum Heart Rate Exploration

In [31]:

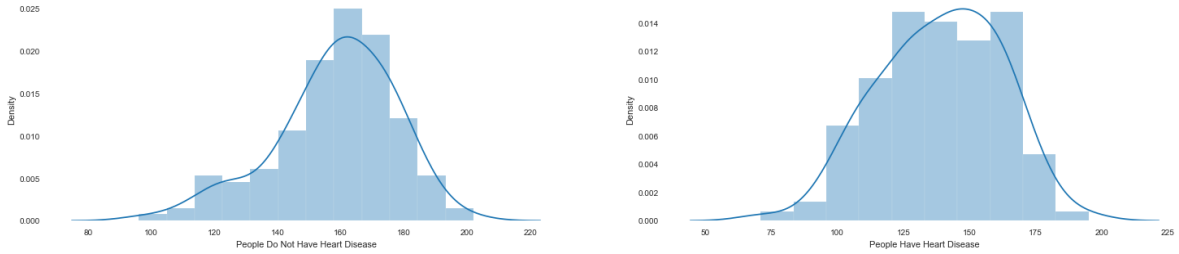
```
# Display maximum heart rate distribution
df['thalach'].plot(kind = 'hist', title = 'Maximum Heart Rate Achieved', figsize = (8, 6))
plt.show()
```



In [32]:

```
# Display maximum heart rate distribution based on heart disease
fig, (axis1, axis2) = plt.subplots(1, 2, figsize=(25, 5))
ax = sns.distplot(df[df['target'] == 1]['thalach'], label='Do not have heart disease', ax =
ax.set(xlabel = 'People Do Not Have Heart Disease')
ax = sns.distplot(df[df['target'] == 2]['thalach'], label = 'Have heart disease', ax = axis
ax.set(xlabel = 'People Have Heart Disease')
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```



In [33]:

```
# Get min, max and average of the maximum heart rate of the people do not have heart diseas
print('Min resting blood pressure of people who do not have heart disease: ', min(df[df['ta
print('Max resting blood pressure of people who do not have heart disease: ', max(df[df['ta
print('Average resting blood pressure of people who do not have heart disease: ', df[df['ta
```

```
Min resting blood pressure of people who do not have heart disease:  96
Max resting blood pressure of people who do not have heart disease:  202
Average resting blood pressure of people who do not have heart disease:  15
8.33333333333334
```

In [34]:

```
# Get min, max and average of the maximum heart rate of the people have heart diseas
print('Min maximum heart rate  of people who have heart disease: ', min(df[df['target'] ==
print('Max maximum heart rate people who have heart disease: ', max(df[df['target'] == 2]['
print('Average maximum heart rate of people who have heart disease: ', df[df['target'] == 2
```

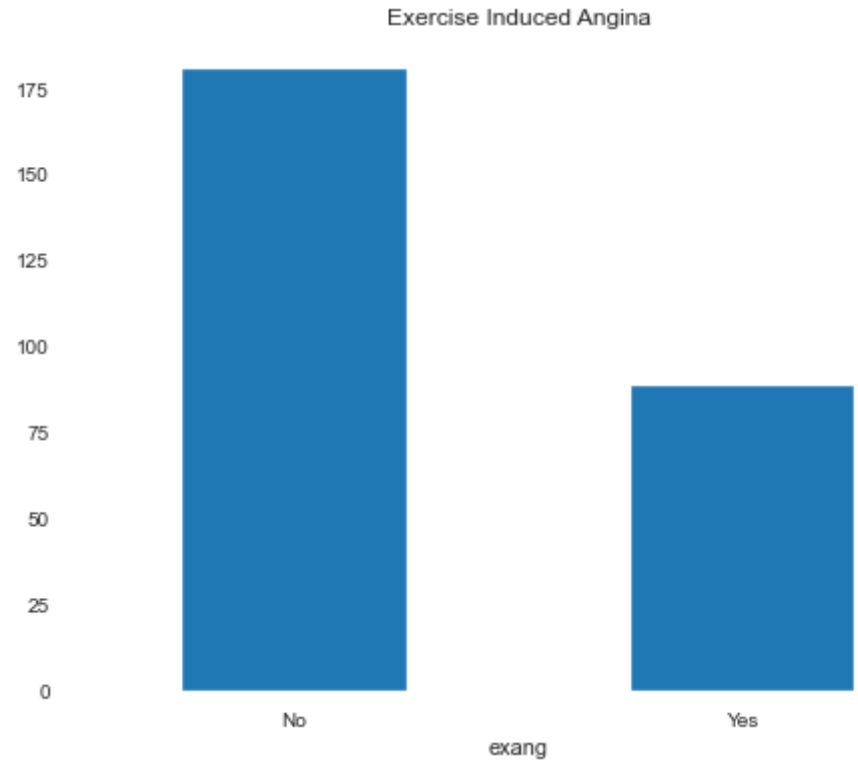
```
Min maximum heart rate  of people who have heart disease:  71
Max maximum heart rate people who have heart disease:  195
Average maximum heart rate of people who have heart disease:  138.8583333333
3332
```

The people who have heigh heart rate greater than 150 are more likely to have heart disease.

Exercise Induced Angina Exploration

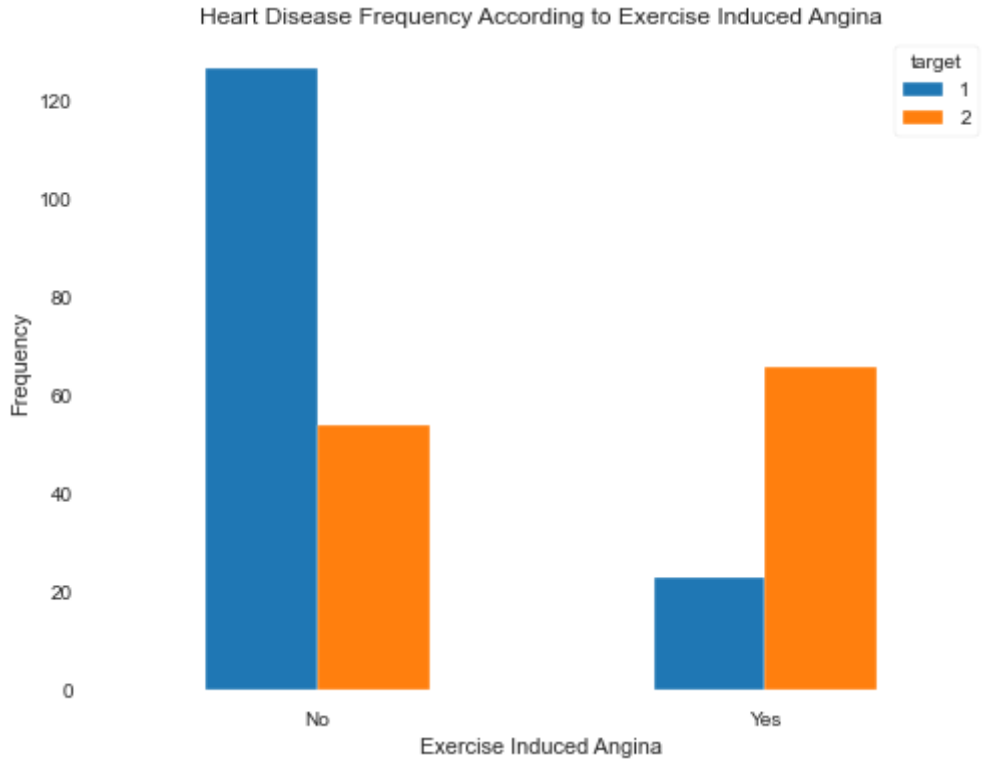
In [35]:

```
# Display exercise induced angina in bar chart
df.groupby(df['exang']).count()['target'].plot(kind = 'bar', title = 'Exercise Induced Angi
plt.xticks(np.arange(2), ('No', 'Yes'), rotation = 0)
plt.show()
```



In [36]:

```
# Display exercise induced angina based on the target
pd.crosstab(df.exang,df.target).plot(kind = "bar", figsize = (8, 6))
plt.title('Heart Disease Frequency According to Exercise Induced Angina')
plt.xlabel('Exercise Induced Angina')
plt.xticks(np.arange(2), ('No', 'Yes'), rotation = 0)
plt.ylabel('Frequency')
plt.show()
```

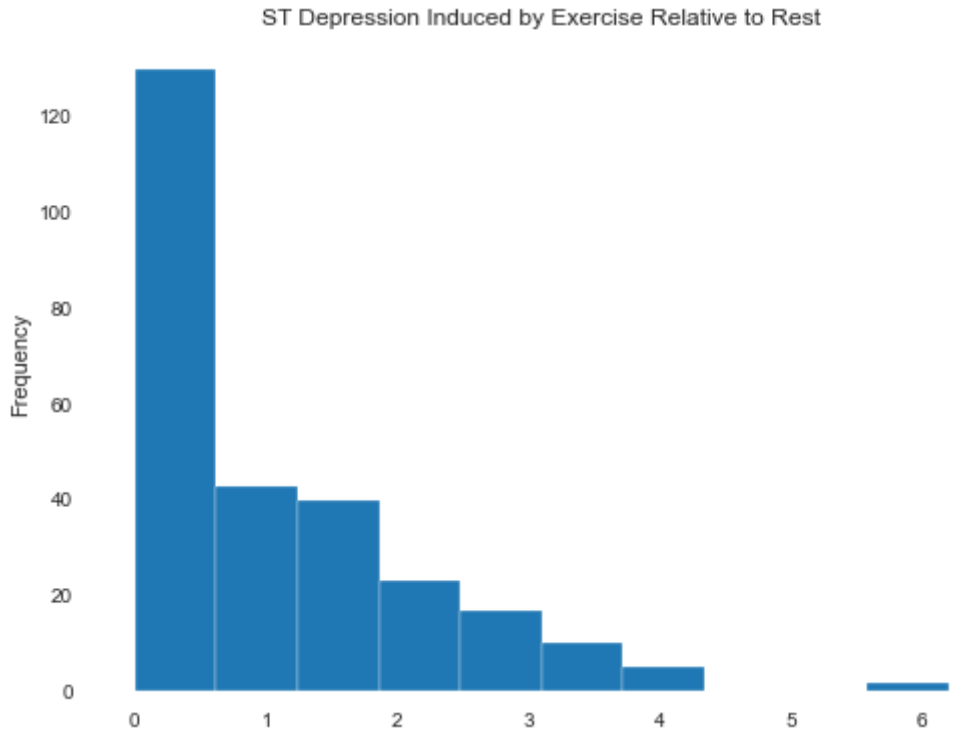


The people who suffer from exercise induced angina are more likely to likely to be infected with the heart disease.

ST depression Exploration

In [37]:

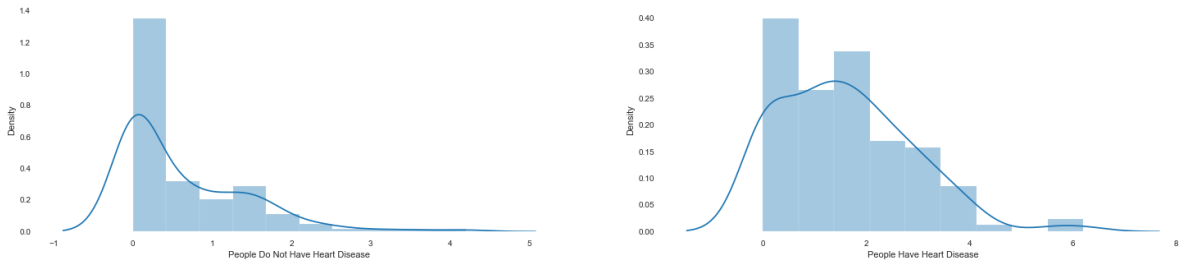
```
# Display ST depression induced by exercise relative to rest distribution
df['oldpeak'].plot(kind = 'hist', title = 'ST Depression Induced by Exercise Relative to Re
plt.show()
```



In [38]:

```
# Display ST depression distribution based on heart disease
fig, (axis1, axis2) = plt.subplots(1, 2, figsize=(25, 5))
ax = sns.distplot(df[df['target'] == 1]['oldpeak'], label='Do not have heart disease', ax =
ax.set(xlabel = 'People Do Not Have Heart Disease')
ax = sns.distplot(df[df['target'] == 2]['oldpeak'], label = 'Have heart disease', ax = axis
ax.set(xlabel = 'People Have Heart Disease')
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```



In [39]:

```
# Get min, max and average of the ST depression  of the people have heart diseas
print('Min ST depression of people who do not have heart disease: ', min(df[df['target'] ==
print('Max ST depression of people who do not have heart disease: ', max(df[df['target'] ==
print('Average ST depression of people who do not have heart disease: ', df[df['target'] ==
```

```
Min ST depression of people who do not have heart disease:  0.0
Max ST depression of people who do not have heart disease:  4.2
Average ST depression of people who do not have heart disease:  0.6226666666
666668
```

In [40]:

```
# Get min, max and average of the ST depression of the people have heart diseas
print('Min ST depression of people who have heart disease: ', min(df[df['target'] == 2]['ol
print('Max ST depression of people who have heart disease: ', max(df[df['target'] == 2]['ol
print('Average ST depression of people not have heart disease: ', df[df['target'] == 2]['ol
```

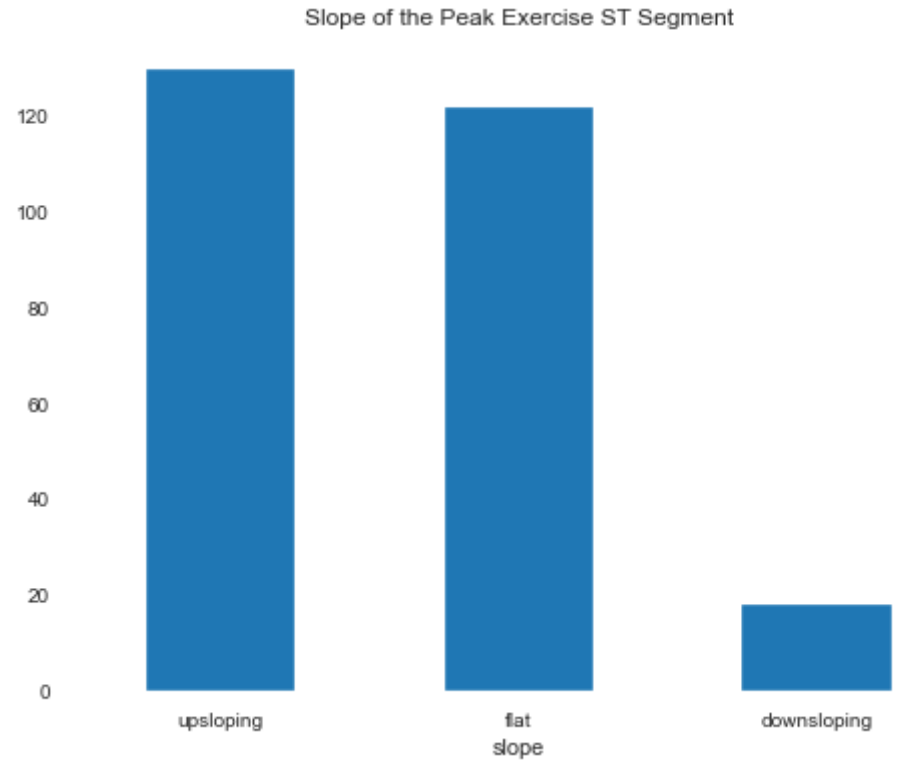
```
Min ST depression of people who have heart disease:  0.0
Max ST depression of people who have heart disease:  6.2
Average ST depression of people not have heart disease:  1.584166666666667
```

The average ST depression of people who do not have heart disease is 0.6 and the average ST depression of people have heart disease is 1.5.

Slope Exploration

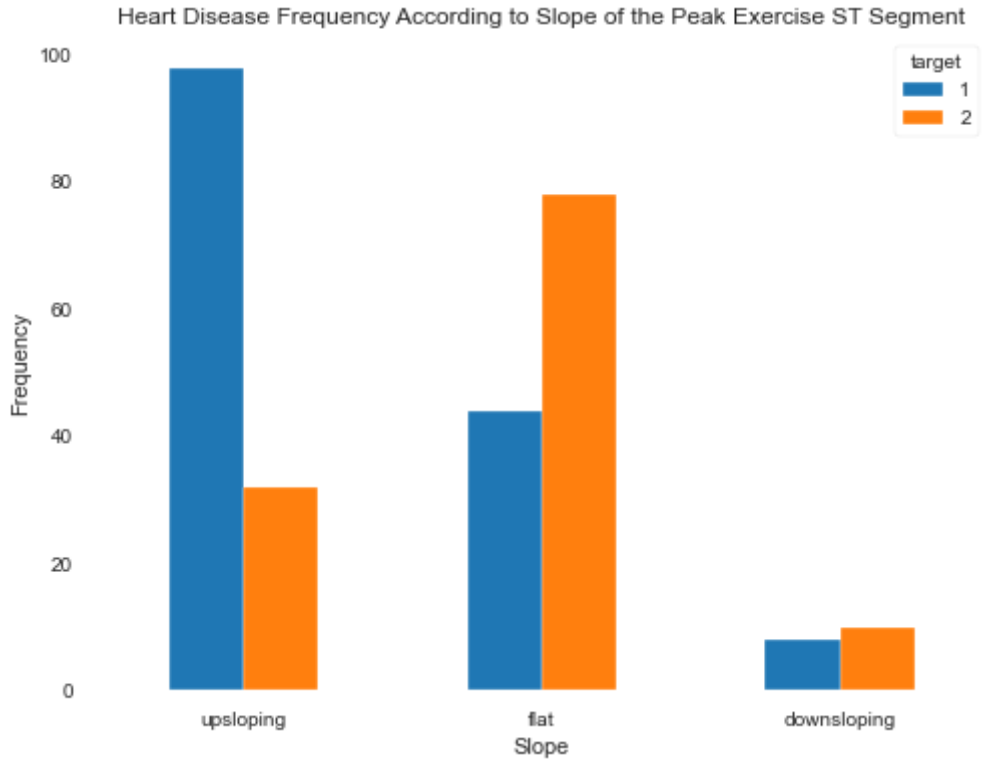
In [41]:

```
# Display slope of the peak exercise ST segment in bar chart
df.groupby(df['slope']).count()['target'].plot(kind = 'bar', title = 'Slope of the Peak Exe
plt.xticks(np.arange(3), ('upsloping', 'flat', 'downsloping'), rotation = 0)
plt.show()
```



In [42]:

```
# Display slope of the peak exercise ST segment based on the target
pd.crosstab(df.slope,df.target).plot(kind = "bar", figsize = (8, 6))
plt.title('Heart Disease Frequency According to Slope of the Peak Exercise ST Segment')
plt.xlabel('Slope')
plt.xticks(np.arange(3), ('upsloping', 'flat', 'downsloping'), rotation = 0)
plt.ylabel('Frequency')
plt.show()
```

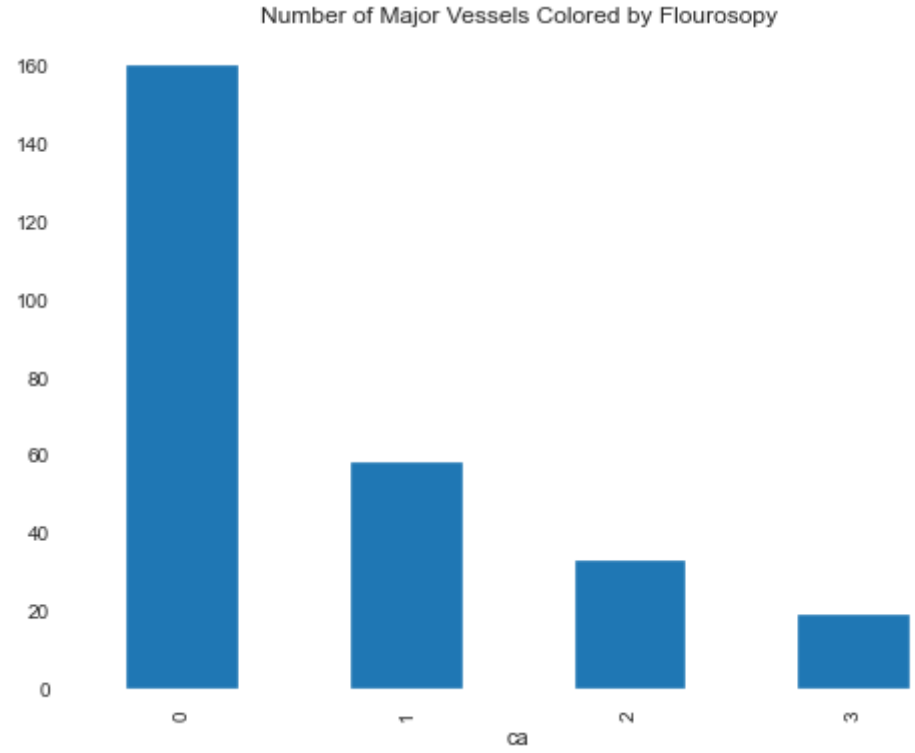


As it is clear, the people with flat peak ST segment are likely to have heart disease and usually the people who do not have heart disease have upsloping peak ST segment.

Major Vessels Exploration

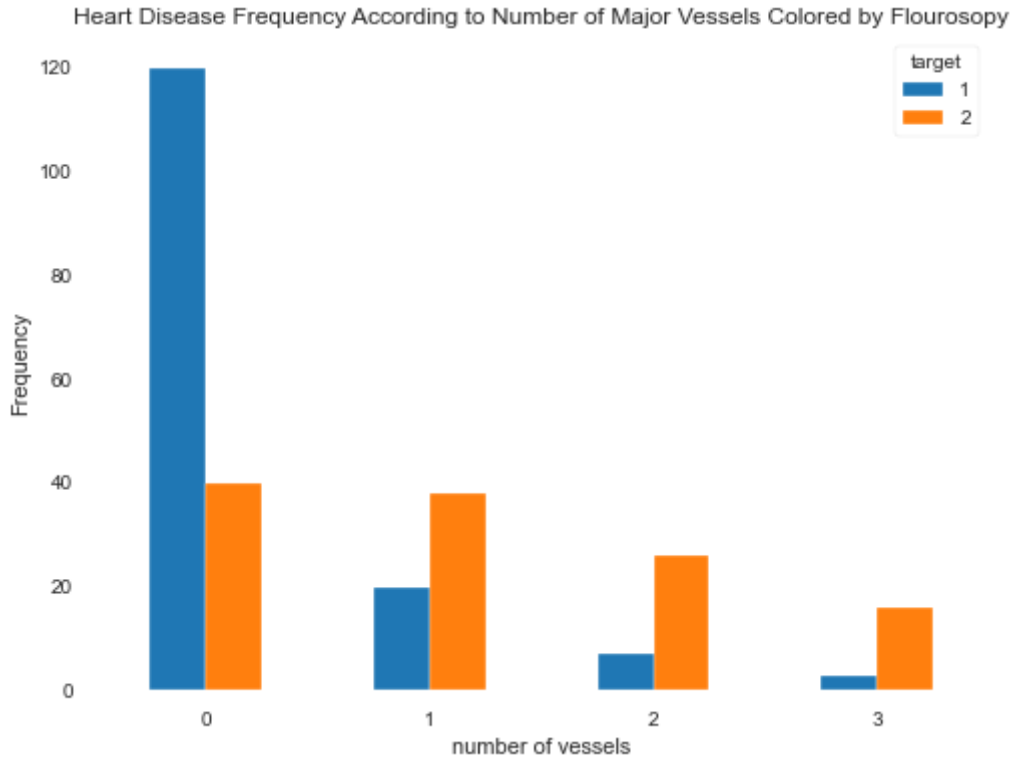
In [43]:

```
# Display number of major vessels in bar chart
df.groupby(df['ca']).count()['target'].plot(kind = 'bar', title = 'Number of Major Vessels
                                            figsize = (8, 6))
plt.show()
```



In [44]:

```
# Display number of vessels based on the target
pd.crosstab(df.ca,df.target).plot(kind = "bar", figsize = (8, 6))
plt.title('Heart Disease Frequency According to Number of Major Vessels Colored by Flouroso
plt.xlabel('number of vessels')
plt.xticks(rotation = 0)
plt.ylabel('Frequency')
plt.show()
```

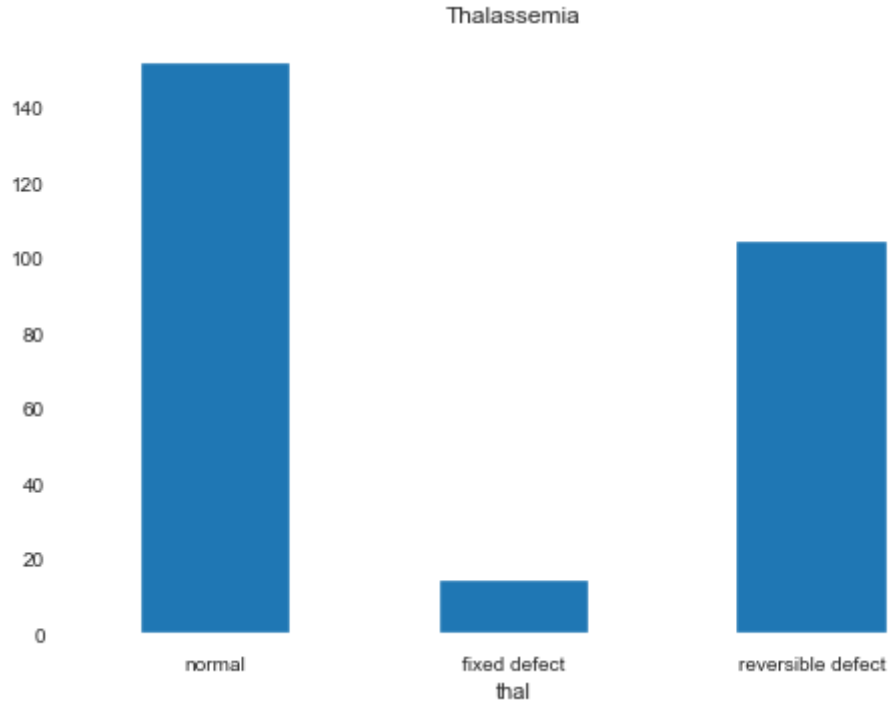


As it is clear, the people who do not have heart disease usually do not have major vessels colored by flourosopy.

Thalassemia Exploration

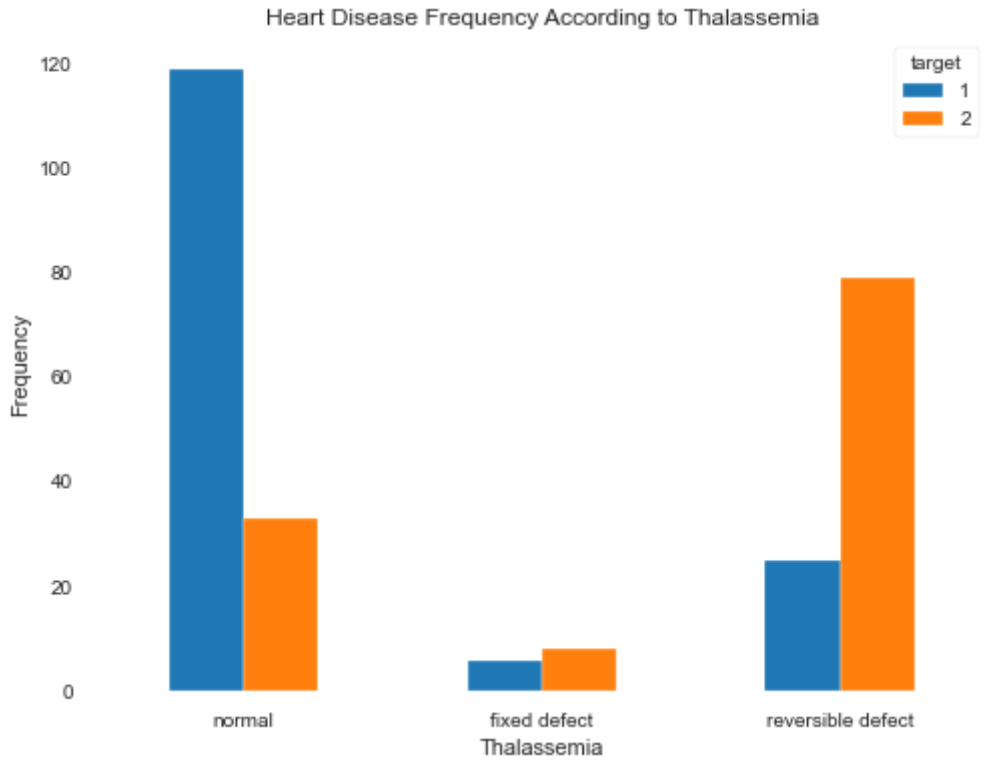
In [45]:

```
# Display thalassemia in bar chart
df.groupby(df['thal']).count()['target'].plot(kind = 'bar', title = 'Thalassemia')
plt.xticks(np.arange(3), ('normal', 'fixed defect', 'reversible defect'), rotation = 0)
plt.show()
```



In [46]:

```
pd.crosstab(df.thal,df.target).plot(kind = "bar", figsize = (8, 6))
plt.title('Heart Disease Frequency According to Thalassemia')
plt.xlabel('Thalassemia')
plt.xticks(np.arange(3), ('normal', 'fixed defect', 'reversible defect'), rotation = 0)
plt.ylabel('Frequency')
plt.show()
```



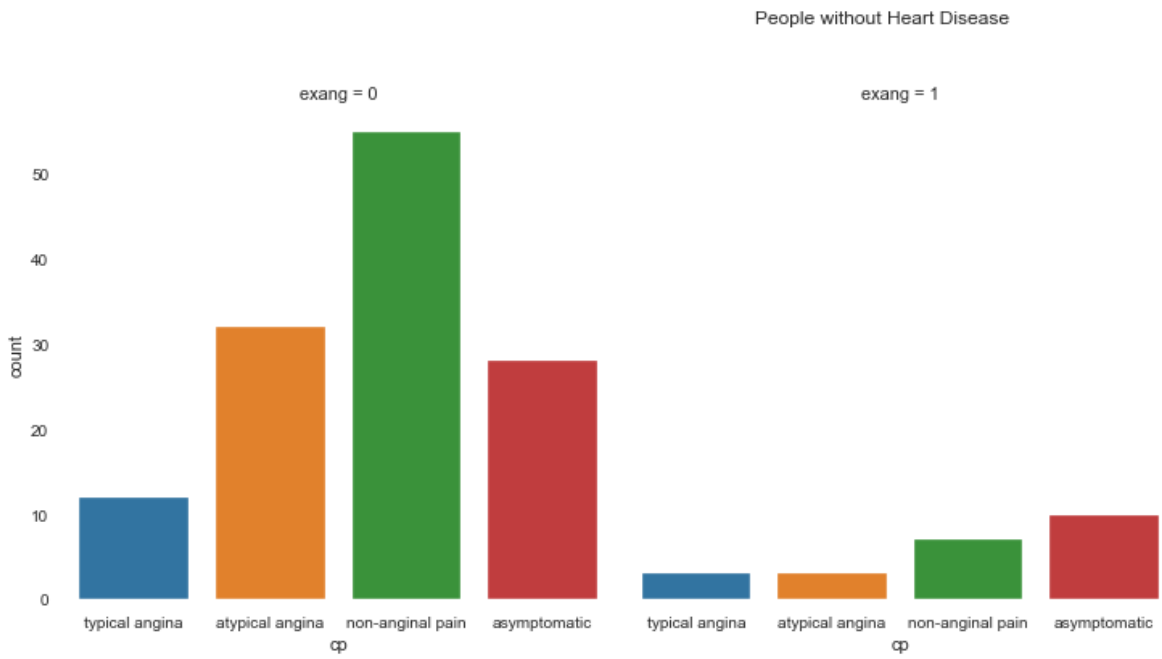
As it is clear, the people with reversible defect are likely to have heart disease.

The correlation between heart disease, cp and exang

In [47]:

```
g = sns.factorplot("cp", col = "exang", col_wrap = 3, data = df[df['target'] == 1], kind =
plt.xticks(np.arange(4), ('typical angina', 'atypical angina', 'non-anginal pain', 'asympto
g.fig.suptitle('People without Heart Disease', y = 1.1)
plt.show()
```

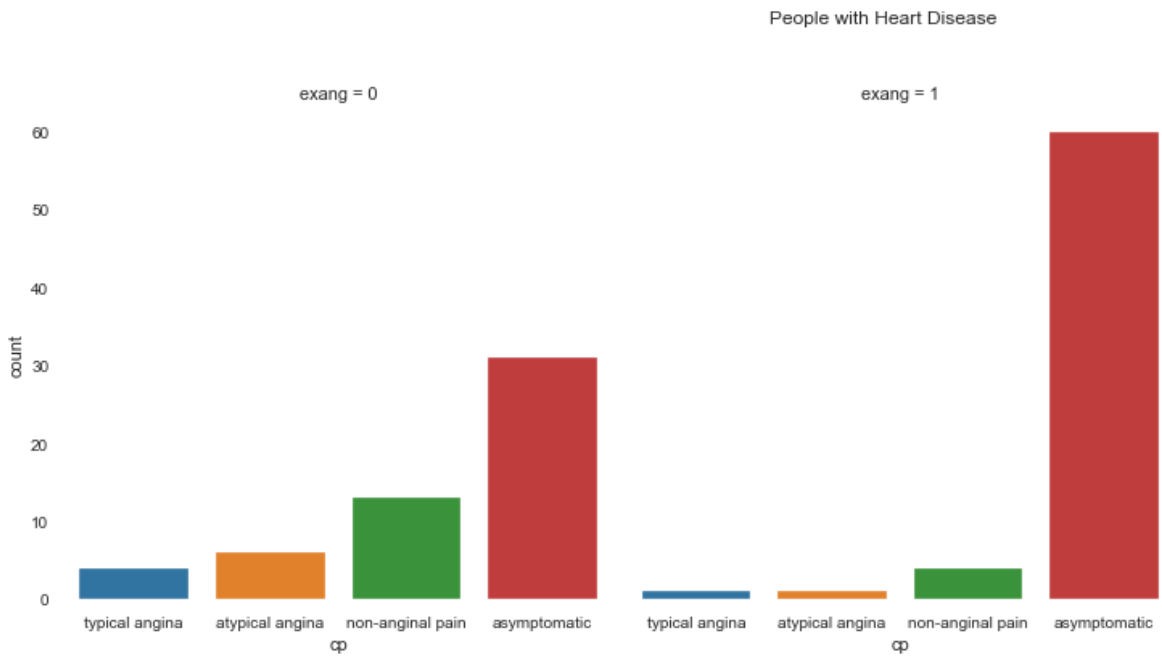
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:3714: User
Warning:

The `factorplot` function has been renamed to `catplot`. The original name w
ill be removed in a future release. Please update your code. Note that the d
efault `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot
`.

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning:

Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments withou
t an explicit keyword will result in an error or misinterpretation.
```



In [48]:

```
g = sns.factorplot("cp", col = "exang", col_wrap = 3, data = df[df['target'] == 2], kind =
plt.xticks(np.arange(4), ('typical angina', 'atypical angina', 'non-anginal pain', 'asympto
g.fig.suptitle('People with Heart Disease', y = 1.1)
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:3714: User
Warning:

The `factorplot` function has been renamed to `catplot`. The original name w
ill be removed in a future release. Please update your code. Note that the d
efault `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot
`.

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning:

Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments withou
t an explicit keyword will result in an error or misinterpretation.
```

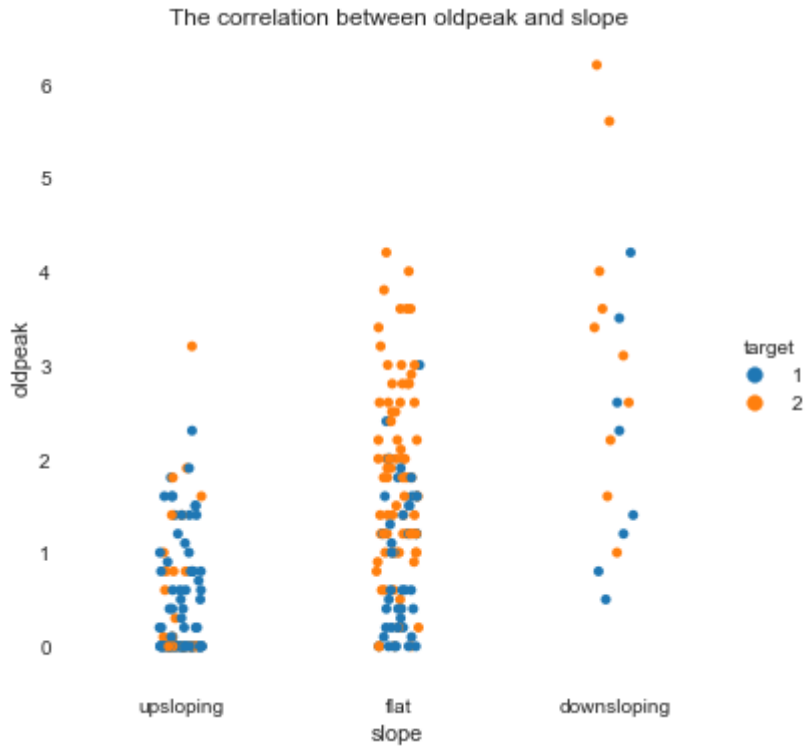


The people who have exercise induced angina, they usually suffer from asymptomatic chest pain. Moreover, the people who do not have exercise induced angina, they usually suffer from asymptomatic chest pain if they have heart disease.

The correlation between oldpeak, slope and target

In [49]:

```
sns.catplot(x = "slope", y = "oldpeak", hue = "target", data = df)
plt.title('The correlation between oldpeak and slope')
plt.xticks(np.arange(3), ('upsloping', 'flat', 'downsloping'), rotation = 0)
plt.show()
```



Flat slope and downsloping have higher values of ST depression.

The correlation between ca and age

In [50]:

```
g = sns.catplot(x = 'ca', y = 'age', hue = 'target', data = df, kind="swarm")
g.fig.suptitle('The correlation between number of major vessels colored by flourosopy and a
plt.show()
```

The older people are more likely to have vessels colored by flourosopy.

The correlation between age and thalach

In [51]:

```
sns.relplot(x = 'age', y = 'thalach', data = df, hue = 'target', legend="full")
plt.title('The correlation between age and heart rate')
plt.show()
```

The older the person, the lower the heart rate. Also, the people with lower heart rate are likely to have heart disease.

3. Modeling

3.1. Prepare Data for Machine Learning

In [52]:

```
# Initialize data and target
target = df['target']
features = df.drop(['target'], axis = 1)
```

In [53]:

```
# Split the data into training set and testing set
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size = 0.2, rand
```

Here, we will use the below machine learning algorithms then I will select the best one based on its

classification report.

- Support Vector Machine
- Random Forest
- Decision Tree
- Extreme Gradient Boosting
- Logistic Regression
- Naive Bayes
- K - Neighbours

3.2. Train and Evaluate Models

In [54]:

```
# Train and evaluate model
def fit_eval_model(model, train_features, y_train, test_features, y_test):

    """
    Function: train and evaluate a machine learning classifier.
    Args:
      model: machine learning classifier
      train_features: train data extracted features
      y_train: train data lables
      test_features: train data extracted features
      y_test: train data lables
    Return:
      results(dictionary): a dictionary of classification report
    """
    results = {}

    # Train the model
    model.fit(train_features, y_train)

    # Test the model
    train_predicted = model.predict(train_features)
    test_predicted = model.predict(test_features)

     # Classification report and Confusion Matrix
    results['classification_report'] = classification_report(y_test, test_predicted)
    results['confusion_matrix'] = confusion_matrix(y_test, test_predicted)

    return results
```

Model Evaluation

In [55]:

```
m1 = 'Logistic Regression'
lr = LogisticRegression()
model = lr.fit(X_train, y_train)
lr_predict = lr.predict(X_test)
lr_conf_matrix = confusion_matrix(y_test, lr_predict)
lr_acc_score = accuracy_score(y_test, lr_predict)
print("confussion matrix")
print(lr_conf_matrix)
print("\n")
print("Accuracy of Logistic Regression:",lr_acc_score*100,'\n')
print(classification_report(y_test,lr_predict))
```

```
confussion matrix
[[26  4]
 [ 5 19]]


Accuracy of Logistic Regression: 83.33333333333334

              precision    recall  f1-score   support

           1       0.84      0.87      0.85        30
           2       0.83      0.79      0.81        24

    accuracy                           0.83        54
   macro avg       0.83      0.83      0.83        54
weighted avg       0.83      0.83      0.83        54


C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:763: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
```

In [56]:

```
m2 = 'Naive Bayes'
nb = GaussianNB()
nb.fit(X_train,y_train)
nbpred = nb.predict(X_test)
nb_conf_matrix = confusion_matrix(y_test, nbpred)
nb_acc_score = accuracy_score(y_test, nbpred)
print("confussion matrix")
print(nb_conf_matrix)
print("\n")
print("Accuracy of Naive Bayes model:",nb_acc_score*100,'\n')
print(classification_report(y_test,nbpred))
```

```
confussion matrix
[[23  7]
 [ 7 17]]


Accuracy of Naive Bayes model: 74.07407407407408

              precision    recall  f1-score   support

           1       0.77      0.77      0.77        30
           2       0.71      0.71      0.71        24

    accuracy                           0.74        54
   macro avg       0.74      0.74      0.74        54
weighted avg       0.74      0.74      0.74        54
```

In [57]:

```
m3 = 'Random Forest Classfier'
rf = RandomForestClassifier(n_estimators=20, random_state=12,max_depth=5)
rf.fit(X_train,y_train)
rf_predicted = rf.predict(X_test)
rf_conf_matrix = confusion_matrix(y_test, rf_predicted)
rf_acc_score = accuracy_score(y_test, rf_predicted)
print("confussion matrix")
print(rf_conf_matrix)
print("\n")
print("Accuracy of Random Forest:",rf_acc_score*100,'\n')
print(classification_report(y_test,rf_predicted))
```

```
confussion matrix
[[28  2]
 [ 6 18]]


Accuracy of Random Forest: 85.18518518518519

              precision    recall  f1-score   support

           1       0.82      0.93      0.87        30
           2       0.90      0.75      0.82        24

    accuracy                           0.85        54
   macro avg       0.86      0.84      0.85        54
weighted avg       0.86      0.85      0.85        54
```

In [58]:

```
m4 = 'Extreme Gradient Boost'
xgb = XGBClassifier(learning_rate=0.1, n_estimators=25, max_depth=15,gamma=0.8, subsample=0
                    reg_lambda=2, booster='dart', colsample_bylevel=0.6, colsample_bynode=0
xgb.fit(X_train, y_train)
xgb_predicted = xgb.predict(X_test)
xgb_conf_matrix = confusion_matrix(y_test, xgb_predicted)
xgb_acc_score = accuracy_score(y_test, xgb_predicted)
print("confussion matrix")
print(xgb_conf_matrix)
print("\n")
print("Accuracy of Extreme Gradient Boost:",xgb_acc_score*100,'\n')
print(classification_report(y_test,xgb_predicted))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarn
ing:

The use of label encoder in XGBClassifier is deprecated and will be removed
in a future release. To remove this warning, do the following: 1) Pass optio
n use_label_encoder=False when constructing XGBClassifier object; and 2) Enc
ode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_cla
ss - 1].


[10:54:27] WARNING: D:\bld\xgboost-split_1645118015404\work\src\learner.cc:1
115: Starting in XGBoost 1.3.0, the default evaluation metric used with the
objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitl
y set eval_metric if you'd like to restore the old behavior.
confussion matrix
[[25  5]
 [ 7 17]]


Accuracy of Extreme Gradient Boost: 77.77777777777779

              precision    recall  f1-score   support

           1       0.78      0.83      0.81        30
           2       0.77      0.71      0.74        24

    accuracy                           0.78        54
   macro avg       0.78      0.77      0.77        54
weighted avg       0.78      0.78      0.78        54
```

In [59]:

```
m5 = 'K-NeighborsClassifier'
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train, y_train)
knn_predicted = knn.predict(X_test)
knn_conf_matrix = confusion_matrix(y_test, knn_predicted)
knn_acc_score = accuracy_score(y_test, knn_predicted)
print("confussion matrix")
print(knn_conf_matrix)
print("\n")
print("Accuracy of K-NeighborsClassifier:",knn_acc_score*100,'\n')
print(classification_report(y_test,knn_predicted))
```

```
confussion matrix
[[23  7]
 [ 9 15]]


Accuracy of K-NeighborsClassifier: 70.37037037037037

              precision    recall  f1-score   support

           1       0.72      0.77      0.74        30
           2       0.68      0.62      0.65        24

    accuracy                           0.70        54
   macro avg       0.70      0.70      0.70        54
weighted avg       0.70      0.70      0.70        54
```

In [60]:

```
m6 = 'DecisionTreeClassifier'
dt = DecisionTreeClassifier(criterion = 'entropy',random_state=0,max_depth = 6)
dt.fit(X_train, y_train)
dt_predicted = dt.predict(X_test)
dt_conf_matrix = confusion_matrix(y_test, dt_predicted)
dt_acc_score = accuracy_score(y_test, dt_predicted)
print("confussion matrix")
print(dt_conf_matrix)
print("\n")
print("Accuracy of DecisionTreeClassifier:",dt_acc_score*100,'\n')
print(classification_report(y_test,dt_predicted))
```

```
confussion matrix
[[23  7]
 [ 7 17]]


Accuracy of DecisionTreeClassifier: 74.07407407407408

              precision    recall  f1-score   support

           1       0.77      0.77      0.77        30
           2       0.71      0.71      0.71        24

    accuracy                           0.74        54
   macro avg       0.74      0.74      0.74        54
weighted avg       0.74      0.74      0.74        54
```

In [61]:

```
m7 = 'Support Vector Classifier'
svc =  SVC(kernel='rbf', C=2)
svc.fit(X_train, y_train)
svc_predicted = svc.predict(X_test)
svc_conf_matrix = confusion_matrix(y_test, svc_predicted)
svc_acc_score = accuracy_score(y_test, svc_predicted)
print("confussion matrix")
print(svc_conf_matrix)
print("\n")
print("Accuracy of Support Vector Classifier:",svc_acc_score*100,'\n')
print(classification_report(y_test,svc_predicted))
```

```
confussion matrix
[[27  3]
 [11 13]]


Accuracy of Support Vector Classifier: 74.07407407407408

              precision    recall  f1-score   support

           1       0.71      0.90      0.79        30
           2       0.81      0.54      0.65        24

    accuracy                           0.74        54
   macro avg       0.76      0.72      0.72        54
weighted avg       0.76      0.74      0.73        54
```

Model Selection

In [62]:

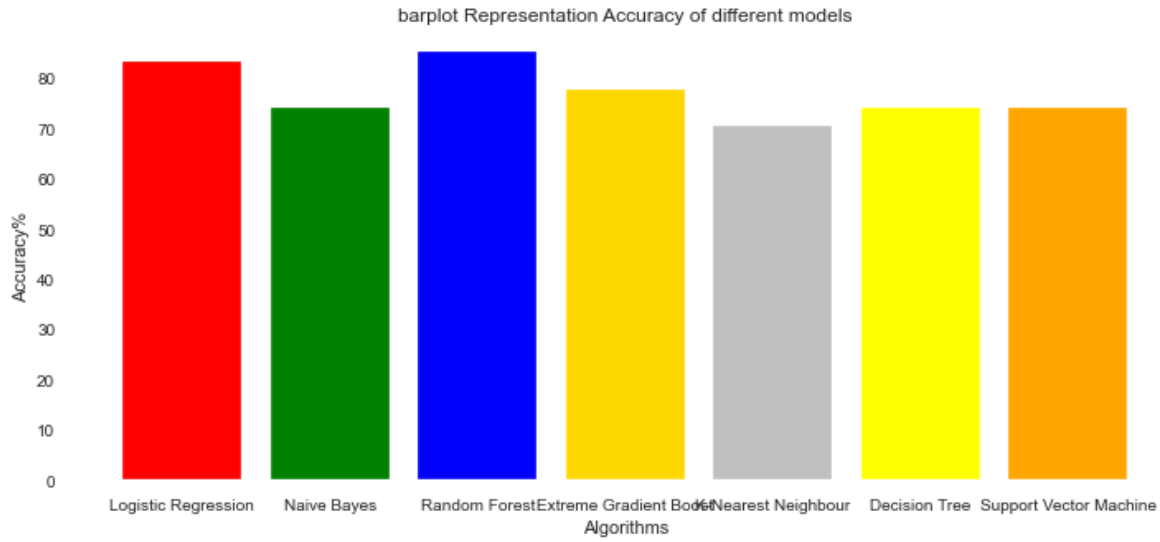
```
model_ev = pd.DataFrame({'Model': ['Logistic Regression','Naive Bayes','Random Forest','Ext
                    'K-Nearest Neighbour','Decision Tree','Support Vector Machine'], 'Accur
                    nb_acc_score*100,rf_acc_score*100,xgb_acc_score*100,knn_acc_score*100,d
model_ev
```

Out[62]:

	Model	Accuracy
0	Logistic Regression	83.333333
1	Naive Bayes	74.074074
2	Random Forest	85.185185
3	Extreme Gradient Boost	77.777778
4	K-Nearest Neighbour	70.370370
5	Decision Tree	74.074074
6	Support Vector Machine	74.074074

In [63]:

```
colors = ['red','green','blue','gold','silver','yellow','orange',]
plt.figure(figsize=(12,5))
plt.title("barplot Representation Accuracy of different models")
plt.xlabel("Algorithms")
plt.ylabel("Accuracy%")
plt.bar(model_ev['Model'],model_ev['Accuracy'],color = colors)
plt.show()
```



3.4. Save Model

Finally, we will save the Random Forest model to use it with WebPage.

In [64]:

```
# Save the model as serialized object pickle
with open('model.pkl', 'wb') as file:
    pickle.dump(rf, file)
```

In [ ]:

```

```