Heart Disease Prediction Using 9 Models

INSPIRATION OF THE PROJECT

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. This research intends to pinpoint the most relevant/risk factors of heart disease as well as predict the overall risk using 9 models([LOGISTIC REGRESSION,KNN, NB,SVM, Random Forest, Decision Tree, XGBoost, GradientBoosting, AdaBoost])

In [ ]:

About Dataset

This is a multivariate type of dataset which means providing or involving a variety of separate mathematical or statistical variables, multivariate numerical data analysis. It is composed of 14 attributes which are age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, oldpeak — ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, number of major vessels and Thalassemia. This database includes 76 attributes, but all published studies relate to the use of a subset of 14 of them. The Cleveland database is the only one used by ML researchers to date. One of the major tasks on this dataset is to predict based on the given attributes of a patient that whether that particular person has heart disease or not and other is the experimental task to diagnose and find out various insights from this dataset which could help in understanding the problem more.

In [ ]:

Aims & Objectives

- we will fill this after some exploratory data analysis

In [ ]:

Import Libraries

- lets start the project by importing all the libraries that we will need in the project.

In [1]:

```
# import libraries

# 1. to handle the data
import pandas as pd
import numpy as np

# 2. To Viusalize the data
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# 3. To preprocess the data
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEnc
from sklearn.impute import SimpleImputer, KNNImputer

# 4. Machine Learning
from sklearn.model_selection import train_test_split

# 5. For Classification task.
from sklearn import datasets, linear_model, metrics

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,G
from xgboost import XGBClassifier

# 6. Metrics
from sklearn.metrics import accuracy_score, confusion_matrix, classifica

# 7. Ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

Load the Dataset¶

In [2]:

```
df = pd.read_csv('heart_disease_uci.csv')
```

In [3]:
```
df
```

Out[3]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalc
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108
2	3	67	Male	Cleveland	asymptomatic	120.0	229.0	False	lv hypertrophy	129
3	4	37	Male	Cleveland	non-anginal	130.0	250.0	False	normal	187
4	5	41	Female	Cleveland	atypical angina	130.0	204.0	False	lv hypertrophy	172
...	...	...	...	...	...	...	...	...	...	
915	916	54	Female	VA Long Beach	asymptomatic	127.0	333.0	True	st-t abnormality	154
916	917	62	Male	VA Long Beach	typical angina	NaN	139.0	False	st-t abnormality	Na
917	918	55	Male	VA Long Beach	asymptomatic	122.0	223.0	True	st-t abnormality	100
918	919	58	Male	VA Long Beach	asymptomatic	NaN	385.0	True	lv hypertrophy	Na
919	920	62	Male	VA Long Beach	atypical angina	120.0	254.0	False	lv hypertrophy	93

920 rows × 16 columns

Exploratory Data Analysis (EDA)¶

Explore Each Column¶

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   id        920 non-null    int64  
 1   age       920 non-null    int64  
 2   sex       920 non-null    object 
 3   dataset   920 non-null    object 
 4   cp        920 non-null    object 
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object 
 8   restecg   918 non-null    object 
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object 
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object 
 13  ca        309 non-null    float64
 14  thal      434 non-null    object 
 15  num       920 non-null    int64  
dtypes: float64(5), int64(3), object(8)
memory usage: 115.1+ KB
```

In [5]:
```
df.shape
```

Out[5]:
```
(920, 16)
```

In [6]:
```
# Minimum age
df['age'].min()
```

Out[6]:
```
28
```

In [7]:
```
# Maximum age

df['age'].max()
```

Out[7]:
```
77
```

Visualizations¶

In [8]:
```
# lets summerize the age column
df['age'].describe()
```

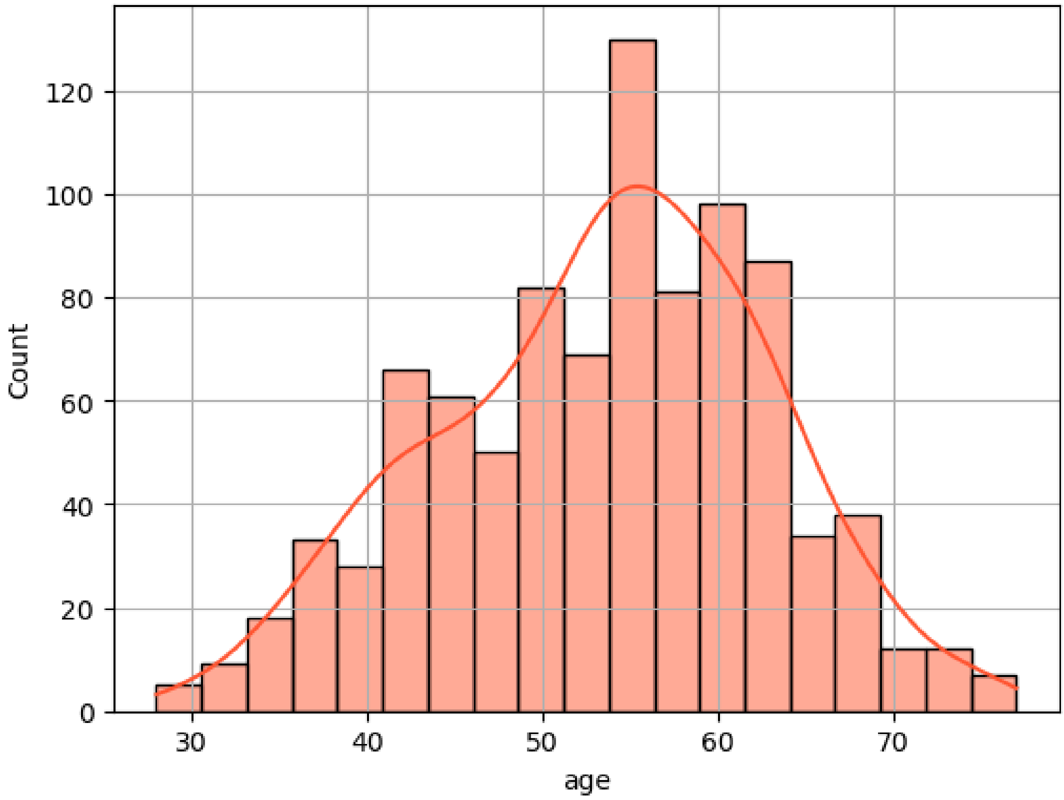
Out[8]:
```
count    920.000000
mean      53.510870
std        9.424685
min       28.000000
25%       47.000000
50%       54.000000
75%       60.000000
max       77.000000
Name: age, dtype: float64
```

- NO missing values in the column so we are good to go...

In [ ]:

In [9]:
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.grid()


# Define custom colors
custom_colors = ["#FF5733", "#3366FF", "#33FF57"]  # Example colors, you

# Plot the histogram with custom colors
sns.histplot(df['age'], kde=True, color="#FF5733", palette=custom_colors
```

Out[9]: <Axes: xlabel='age', ylabel='Count'>

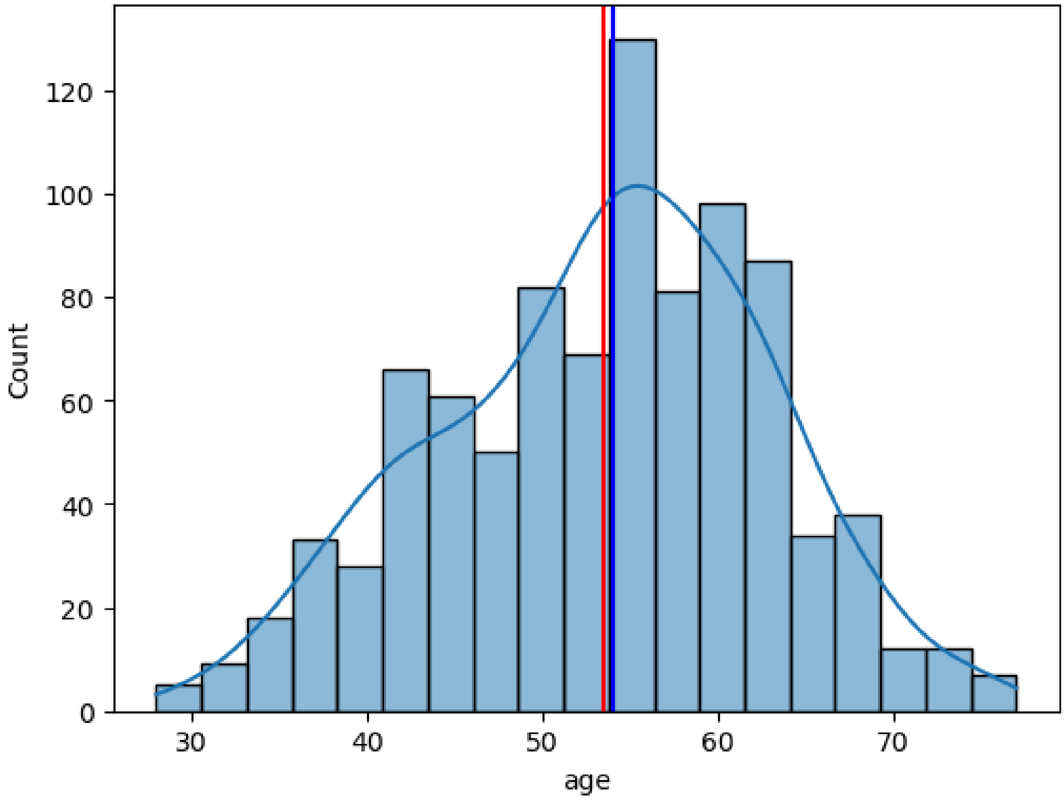


- The age column distribution seems to be normaly distributed because we can clearly see the bill curve.

In [10]:

```
# Plot the mean, Median and mode of age column using sns
sns.histplot(df['age'], kde=True)
plt.axvline(df['age'].mean(), color='Red')
plt.axvline(df['age'].median(), color= 'Green')
plt.axvline(df['age'].mode()[0], color='Blue')

# print the value of mean, median and mode of age column
print('Mean', df['age'].mean())
print('Median', df['age'].median())
print('Mode', df['age'].mode())
```

```
Mean 53.51086956521739
Median 54.0
Mode 0    54
Name: age, dtype: int64
```

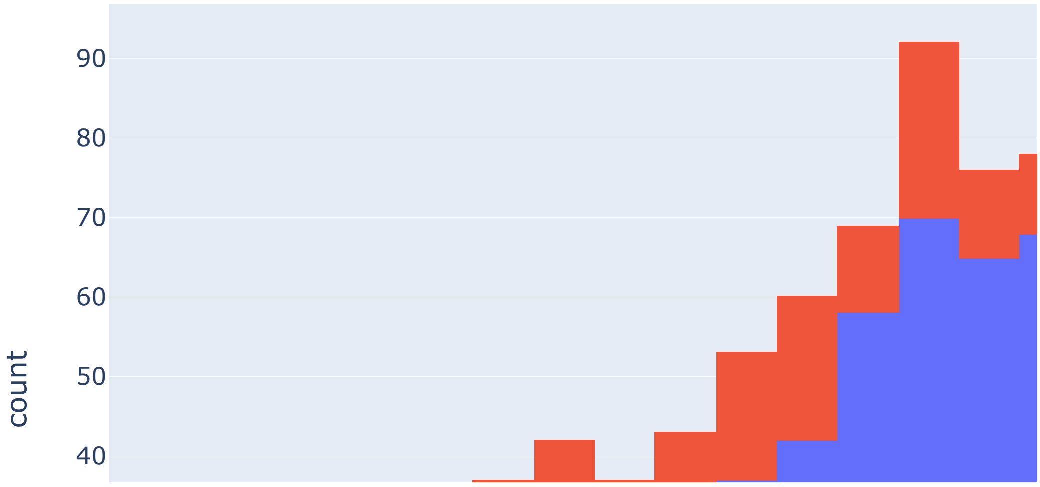


Lets explore the gender base distribution of the dataset for age column

In [ ]:

In [11]:

```
# plot the histogram of age column using plotly and coloring this by sex
import plotly.express as px

fig = px.histogram(data_frame=df, x='age', color= 'sex')
fig.show()
```



Most of the males and females get are with heart disease at the age of 54 to 55 years.

In [ ]:

In [12]:

```
# Find the values of sex column

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

Out[12]:

```
Male      726
Female    194
Name: sex, dtype: int64
```

In [13]:

```python
# calculating the percentage of male and female value counts in the data

male_count = 726
female_count = 194

total_count = male_count + female_count

# Calculating percntage

male_percent = (male_count/total_count)*100
female_percent = (female_count/total_count)*100

# display the results
print(f'Male percentage in the data: {male_percent:.2f}%')
print(f'Female percentage in the data : {female_percent:.2f}%')
```

```
Male percentage in the data: 78.91%
Female percentage in the data : 21.09%
```

In [14]:

```python
# Find the values count of age column grouping by sex column
df.groupby('sex')['age'].value_counts()
```

Out[14]:

```
sex     age
Female  54     15
        51     11
        62     10
        43      9
        48      9
               ..
Male    77      2
        28      1
        31      1
        33      1
        76      1
Name: age, Length: 91, dtype: int64
```

In [15]:

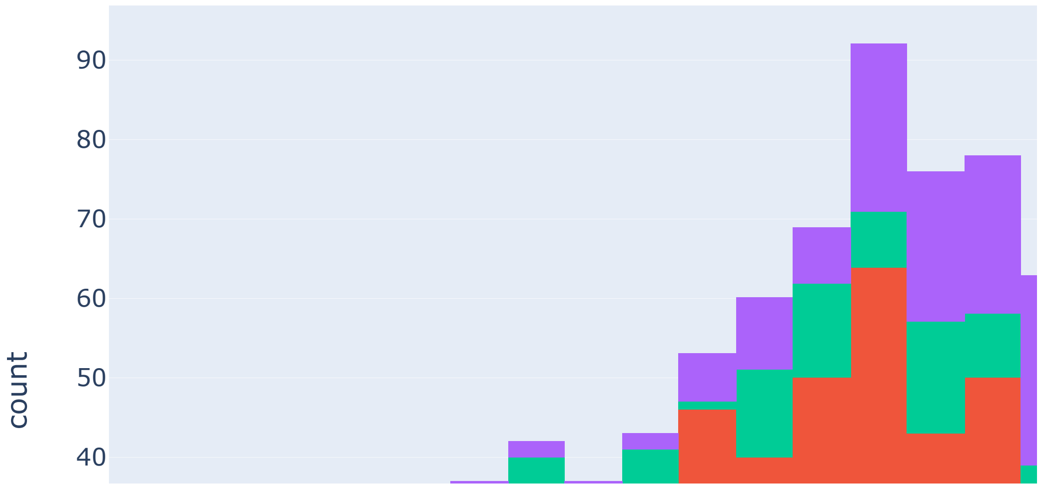
```python
# find the unique values in the dataset column
df['dataset'].value_counts()
```

Out[15]:

```
Cleveland        304
Hungary          293
VA Long Beach    200
Switzerland      123
Name: dataset, dtype: int64
```

In [16]:

```
# make a plot of age column using plotly and coloring by dataset

fig = px.histogram(data_frame=df, x='age', color= 'dataset')
fig.show()

# print the mean median and mode of age column grouped by dataset columr
print("_____________________________________________________________")
print ("Mean of the dataset: ",df.groupby('dataset')['age'].mean())
print("_____________________________________________________________")
print ("Median of the dataset: ",df.groupby('dataset')['age'].median())
print("_____________________________________________________________")
print ("Mode of the dataset: ",df.groupby('dataset')['age'].agg(pd.Serie
print("_____________________________________________________________")
```



```
Mean of the dataset:  dataset
Cleveland        54.351974
Hungary          47.894198
Switzerland      55.317073
VA Long Beach    59.350000
Name: age, dtype: float64
```

```
Median of the dataset:  dataset
Cleveland        55.5
Hungary          49.0
Switzerland      56.0
VA Long Beach    60.0
Name: age, dtype: float64
```

```
Mode of the dataset:  dataset
Cleveland              58
Hungary                54
Switzerland            61
VA Long Beach    [62, 63]
Name: age, dtype: object
```

Exploring CP (Chest Pain) column¶

In [17]:

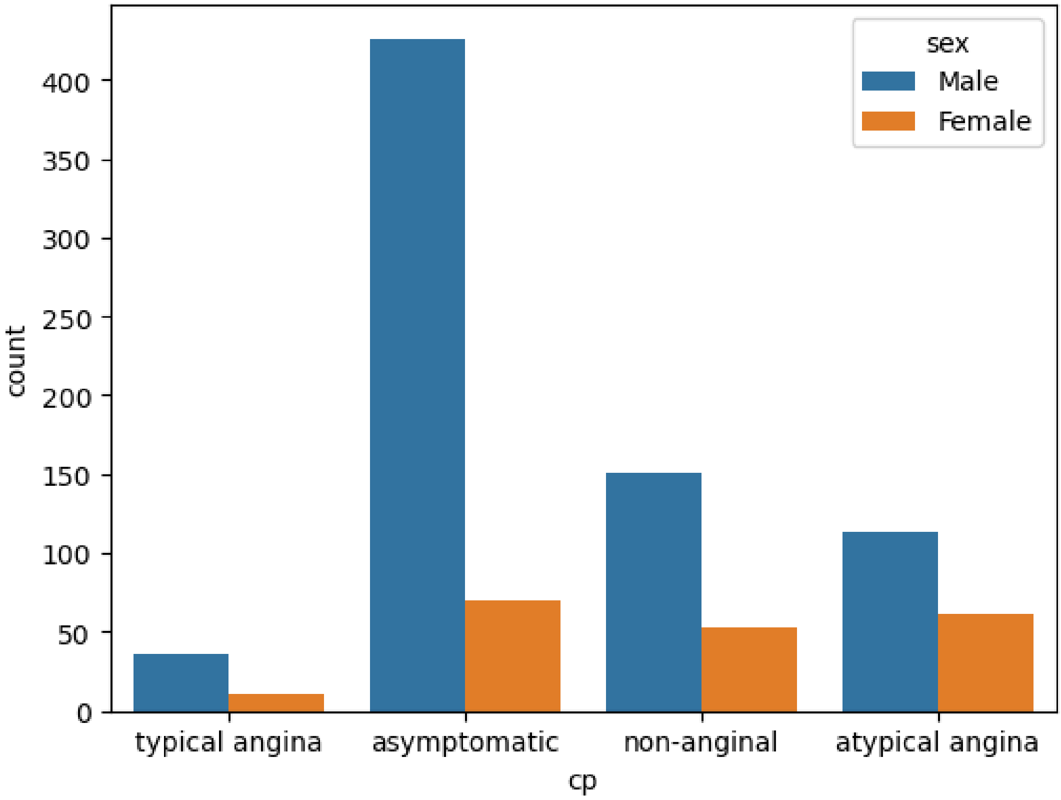
```python
# value count of cp column

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

Out[17]:

```
asymptomatic       496
non-anginal        204
atypical angina    174
typical angina      46
Name: cp, dtype: int64
```

In [18]:
```
# count plot of cp column by sex column

sns.countplot(df, x='cp', hue= 'sex')
```

Out[18]: <Axes: xlabel='cp', ylabel='count'>

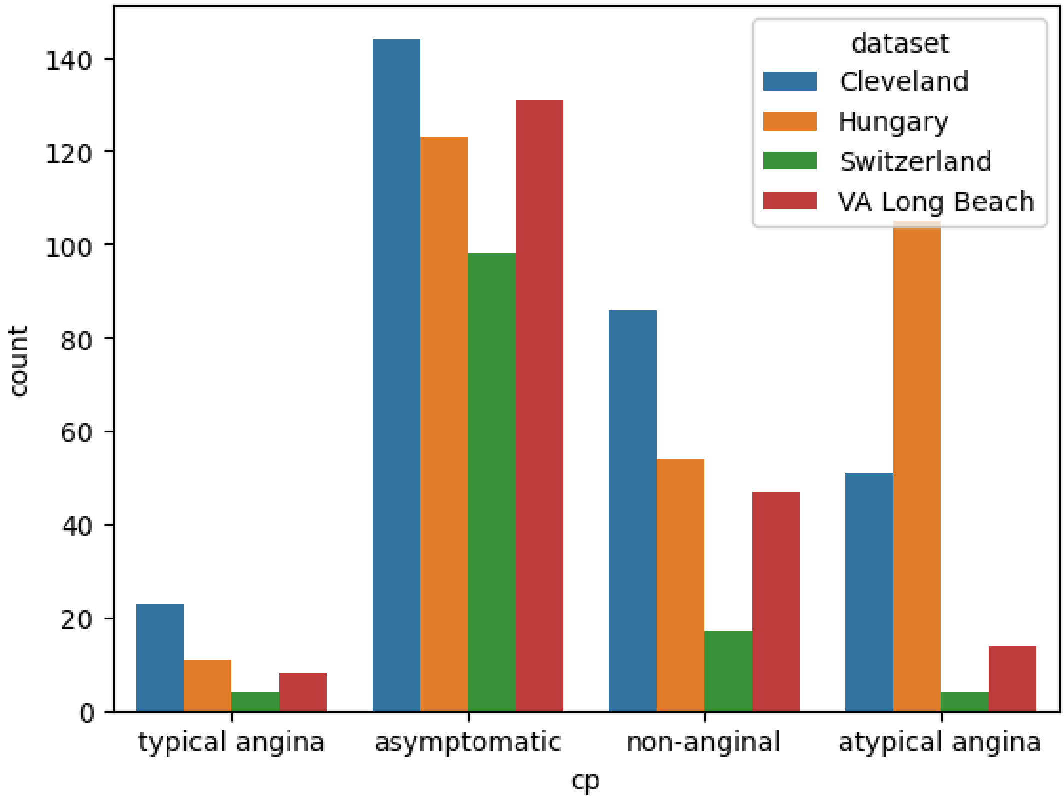


- It shows that maximum male people are getting chest pain and Maximum male people are falling ill in case asymptomatic and very less female people are falling ill in case of typical angina

In [ ]:

In [19]:
```
# count plot of cp column by dataset column
sns.countplot(df,x='cp',hue='dataset')
```

Out[19]: <Axes: xlabel='cp', ylabel='count'>

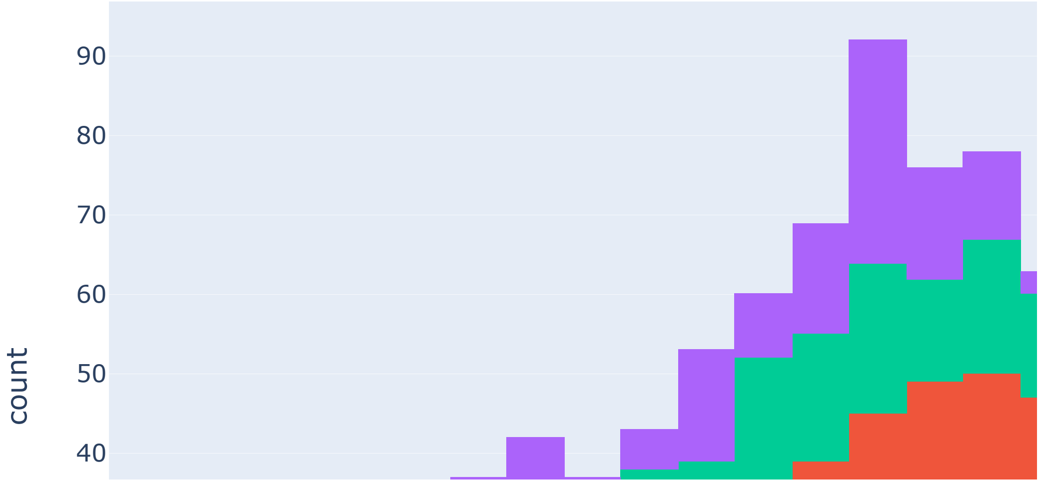


- It shows that Maximum male people getting chest pain in Country Cleveland in case of asymptomatic

8. The high number of Typical angina, Asymptomatic and Non anginal chest pain is in the Cleveland while Atypical anigna is highly occured in Hungary.
9. Lowest number of chest pain (Typical angina, Asymptomatic, Non anginal and Atypical angina)is happened in Switzerland as compare to other origins.

In [20]:

```
# Draw the plot of age column group by cp column

fig = px.histogram(data_frame=df, x='age', color='cp')
fig.show()
```



12. TThe highest number of case of chest pain is happened in 'Asymtomatic Angina' is 45 and the lowest number of chest pain is that happened is Typical Angina is 11.

- The highest number of case of 'Typical Angina' occurred among individuals between the ages of 62 and 63. Notably, 6 individuals within this age range were identified as having Typical Angina.
- The highest number of case of 'Asymtomatic Angina' occurred among individuals between the ages of 56 to 57 years. Notably, 47 individuals within this age group were identified as having Asymptomatic Angina.
- The highest number of case of 'Non Anginal' occurred among individuals between the ages of 54 to 55 years. Notably, 19 individuals within this age group were identified as having Non Anginal.
- The highest number of case of 'Atypical Angina' occurred among individuals between the ages of 54 to 55 years. Notably, 28 individuals within this age group were identified as having Atypical Anginal

In [ ]:

```

```

In [ ]:

```

```

Let's explore the trestbps (resting blood pressure) column:¶

The normal resting blood pressure is 120/80 mm Hg.\ Write here, what will happen if the blood pressure is high or low and then you canbin the data based on the those values.

In [21]:

```
# lets summerize the trestbps column
df['trestbps'].describe()
```

Out[21]:

```
count    861.000000
mean     132.132404
std       19.066070
min        0.000000
25%      120.000000
50%      130.000000
75%      140.000000
max      200.000000
Name: trestbps, dtype: float64
```

Handling missing values in trestbps column¶

There are some missing values becuase total values is 920 but here we have 861

In [22]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   id        920 non-null    int64  
 1   age       920 non-null    int64  
 2   sex       920 non-null    object 
 3   dataset   920 non-null    object 
 4   cp        920 non-null    object 
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object 
 8   restecg   918 non-null    object 
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object 
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object 
 13  ca        309 non-null    float64
 14  thal      434 non-null    object 
 15  num       920 non-null    int64  
dtypes: float64(5), int64(3), object(8)
memory usage: 115.1+ KB
```

Checking the null or blank values

In [23]:
```
df.isnull().sum()
```
Out[23]:
```
id            0
age           0
sex           0
dataset       0
cp            0
trestbps     59
chol         30
fbs          90
restecg       2
thalch       55
exang        55
oldpeak      62
slope       309
ca          611
thal        486
num           0
dtype: int64
```

Filling Numerical Missing values with mean for selected column

Columns are selected based on data types (floating data type)

In [24]:
```
trestbps_mean=df['trestbps'].mean()
df['trestbps'].fillna(trestbps_mean,inplace=True)
```

In [25]:
```
chol_mean=df['chol'].mean()
df['chol'].fillna(chol_mean,inplace=True)
```

In [26]:
```
thalch_mean=df['thalch'].mean()
df['thalch'].fillna(thalch_mean,inplace=True)
```

In [27]:
```
oldpeak_mean=df['oldpeak'].mean()
df['oldpeak'].fillna(oldpeak_mean,inplace=True)
```

In [28]:
```
ca_mean=df['ca'].mean()
df['ca'].fillna(ca_mean,inplace=True)
```

In [29]:
```
df.isnull().sum()
```

Out[29]:
```
id            0
age           0
sex           0
dataset       0
cp            0
trestbps      0
chol          0
fbs          90
restecg       2
thalch        0
exang        55
oldpeak       0
slope       309
ca            0
thal        486
num           0
dtype: int64
```

Fillng Missing categorical columns

In [30]:
```
missing_values =  df.columns[(df.isnull().sum() > 0)]

missing_values
```

Out[30]:
```
Index(['fbs', 'restecg', 'exang', 'slope', 'thal'], dtype='object')
```

In [31]:
```
from sklearn.impute import SimpleImputer

Si = SimpleImputer(strategy='most_frequent')

df['fbs'] = Si.fit_transform(df[['fbs']])

df['exang'] = Si.fit_transform(df[['exang']])

df['slope'] = Si.fit_transform(df[['slope']])

df['thal'] = Si.fit_transform(df[['thal']])

df['restecg'] = Si.fit_transform(df[['restecg']])
```

In [32]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   id        920 non-null    int64  
 1   age       920 non-null    int64  
 2   sex       920 non-null    object 
 3   dataset   920 non-null    object 
 4   cp        920 non-null    object 
 5   trestbps  920 non-null    float64
 6   chol      920 non-null    float64
 7   fbs       920 non-null    object 
 8   restecg   920 non-null    object 
 9   thalch    920 non-null    float64
 10  exang     920 non-null    object 
 11  oldpeak   920 non-null    float64
 12  slope     920 non-null    object 
 13  ca        920 non-null    float64
 14  thal      920 non-null    object 
 15  num       920 non-null    int64  
dtypes: float64(5), int64(3), object(8)
memory usage: 115.1+ KB
```

In [33]:

```
df
```

Out[33]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	
0	1	63	Male	Cleveland	typical angina	145.000000	233.0	True	lv hypertrophy	15
1	2	67	Male	Cleveland	asymptomatic	160.000000	286.0	False	lv hypertrophy	10
2	3	67	Male	Cleveland	asymptomatic	120.000000	229.0	False	lv hypertrophy	12
3	4	37	Male	Cleveland	non-anginal	130.000000	250.0	False	normal	18
4	5	41	Female	Cleveland	atypical angina	130.000000	204.0	False	lv hypertrophy	17
...	...	...	...	...	...	...	...	...	...	
915	916	54	Female	VA Long Beach	asymptomatic	127.000000	333.0	True	st-t abnormality	15
916	917	62	Male	VA Long Beach	typical angina	132.132404	139.0	False	st-t abnormality	13
917	918	55	Male	VA Long Beach	asymptomatic	122.000000	223.0	True	st-t abnormality	10
918	919	58	Male	VA Long Beach	asymptomatic	132.132404	385.0	True	lv hypertrophy	13
919	920	62	Male	VA Long Beach	atypical angina	120.000000	254.0	False	lv hypertrophy	9

920 rows × 16 columns

Encoding all categorical column to numerics Ex-0,1,2

In [34]:
```
from sklearn.preprocessing import LabelEncoder

categorical_cols = df.select_dtypes(include='object').columns

le = LabelEncoder()

for col in categorical_cols:
    df[col] = le.fit_transform(df[col].astype(str))

print(df)
```

```
      id  age  sex  dataset  cp    trestbps   chol  fbs  restecg  
thalch  \
0      1   63    1        0   3  145.000000  233.0    1        0  150.
000000
1      2   67    1        0   0  160.000000  286.0    0        0  108.
000000
2      3   67    1        0   0  120.000000  229.0    0        0  129.
000000
3      4   37    1        0   2  130.000000  250.0    0        1  187.
000000
4      5   41    0        0   1  130.000000  204.0    0        0  172.
000000
..   ...  ...  ...      ...  ..         ...    ...  ...      ...  
...
915  916   54    0        3   0  127.000000  333.0    1        2  154.
000000
916  917   62    1        3   3  132.132404  139.0    0        2  137.
545665
917  918   55    1        3   0  122.000000  223.0    1        2  100.
000000
918  919   58    1        3   0  132.132404  385.0    1        0  137.
545665
919  920   62    1        3   1  120.000000  254.0    0        0   93.
000000

     exang   oldpeak  slope        ca  thal  num
0        0  2.300000      0  0.000000     0    0
1        1  1.500000      1  3.000000     1    2
2        1  2.600000      1  2.000000     2    1
3        0  3.500000      0  0.000000     1    0
4        0  1.400000      2  0.000000     1    0
..     ...       ...    ...       ...   ...  ...
915      0  0.000000      1  0.676375     1    1
916      0  0.878788      1  0.676375     1    0
917      0  0.000000      1  0.676375     0    2
918      0  0.878788      1  0.676375     1    0
919      1  0.000000      1  0.676375     1    1

[920 rows x 16 columns]
```

#separating feature and response/target

X = feature , y = target

In [35]:
```
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

Splitting the data into Training & Test Data

In [36]:
```
from sklearn.model_selection import train_test_split
```

In [37]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_
```

In [38]:
```
X_test
```

Out[38]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak
138	139	35	1	0	0	120.0	198.0	0	1	130.0	1	1.6
766	767	59	1	3	0	122.0	233.0	0	1	117.0	1	1.3
70	71	65	0	0	2	155.0	269.0	0	1	148.0	0	0.8
401	402	48	1	1	1	130.0	245.0	0	1	160.0	0	0.0
754	755	52	1	3	2	122.0	0.0	0	1	110.0	1	2.0
...	...	...	...	...	...	...	...	...	...	...	...	...
34	35	44	1	0	2	130.0	233.0	0	1	179.0	1	0.4
228	229	54	1	0	0	110.0	206.0	0	0	108.0	1	0.0
215	216	56	1	0	3	120.0	193.0	0	0	162.0	0	1.9
409	410	49	1	1	2	140.0	187.0	0	1	172.0	0	0.0
666	667	58	1	2	0	115.0	0.0	0	1	138.0	0	0.5

184 rows × 15 columns

In [39]:
```
y_test
```

Out[39]:
```
138    1
766    1
70     0
401    0
754    2
      ..
34     0
228    3
215    0
409    0
666    1
Name: num, Length: 184, dtype: int64
```

In [40]:
```
#selecting numerical columns
colname=X.select_dtypes(['int64','float64']).columns
colname
```

Out[40]:
```
Index(['id', 'age', 'trestbps', 'chol', 'thalch', 'oldpeak', 'ca'], dt
ype='object')
```

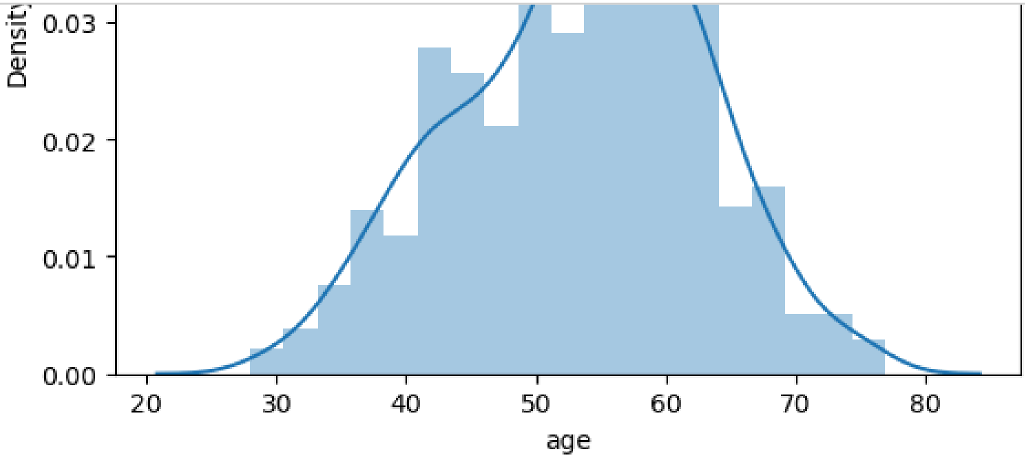
Checking skewness

In [41]:

```
from scipy.stats import skew
```

In [42]:

```
#using for loop to check skewness of all the numeric columns
for col in X[colname]:
    print(col)    #prints column name
    print(skew(X[col])) #prints skewness of col
    plt.figure()
    sns.distplot(X[col])
    plt.show()
```




```
trestbps
0.22013836318121346
```

Its shows all are normal

In [43]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   id        920 non-null    int64  
 1   age       920 non-null    int64  
 2   sex       920 non-null    int32  
 3   dataset   920 non-null    int32  
 4   cp        920 non-null    int32  
 5   trestbps  920 non-null    float64
 6   chol      920 non-null    float64
 7   fbs       920 non-null    int32  
 8   restecg   920 non-null    int32  
 9   thalch    920 non-null    float64
 10  exang     920 non-null    int32  
 11  oldpeak   920 non-null    float64
 12  slope     920 non-null    int32  
 13  ca        920 non-null    float64
 14  thal      920 non-null    int32  
 15  num       920 non-null    int64  
dtypes: float64(5), int32(8), int64(3)
memory usage: 86.4 KB
```

In [44]:
```
for col in df[['id','age','sex','dataset','cp','trestbps','chol','fbs','
               ,'slope','ca','thal','num']]:
     print(df[col])
```

```
3      37
4      41
       ..
915    54
916    62
917    55
918    58
919    62
Name: age, Length: 920, dtype: int64
0      1
1      1
2      1
3      1
4      0
      ..
915    0
916    1
917    1
918    1
919    1
```

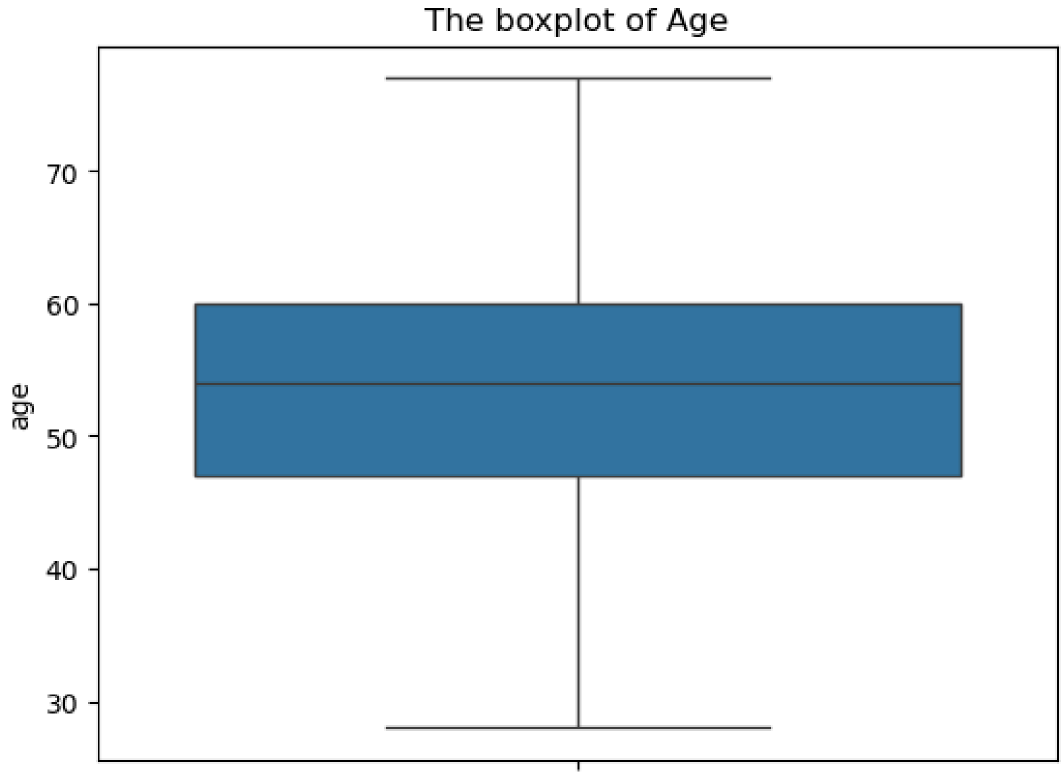
In [45]:
```
df['chol']
```

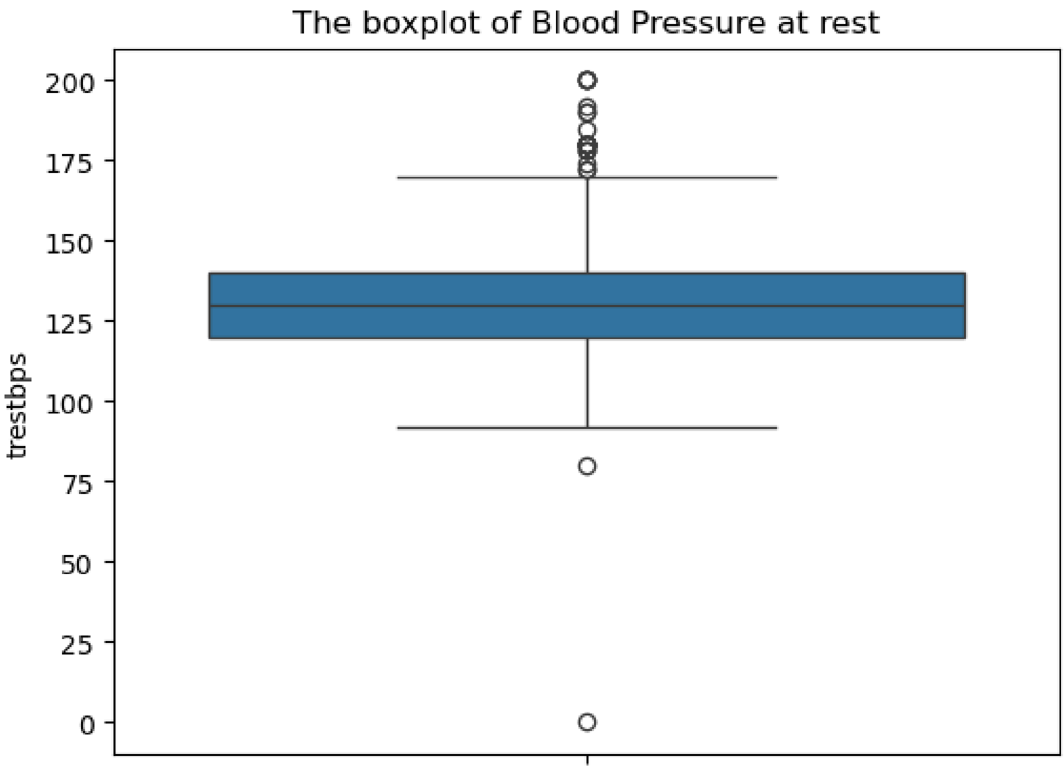
Out[45]:
```
0      233.0
1      286.0
2      229.0
3      250.0
4      204.0
       ...
915    333.0
916    139.0
917    223.0
918    385.0
919    254.0
Name: chol, Length: 920, dtype: float64
```

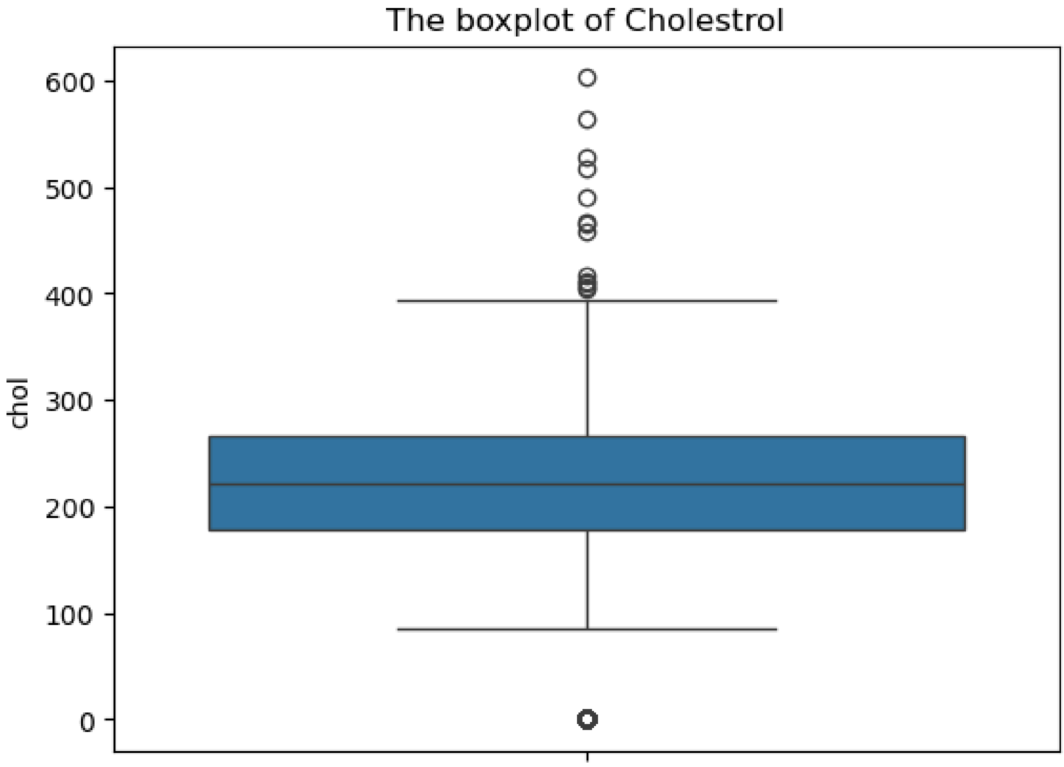
Checking Outliers

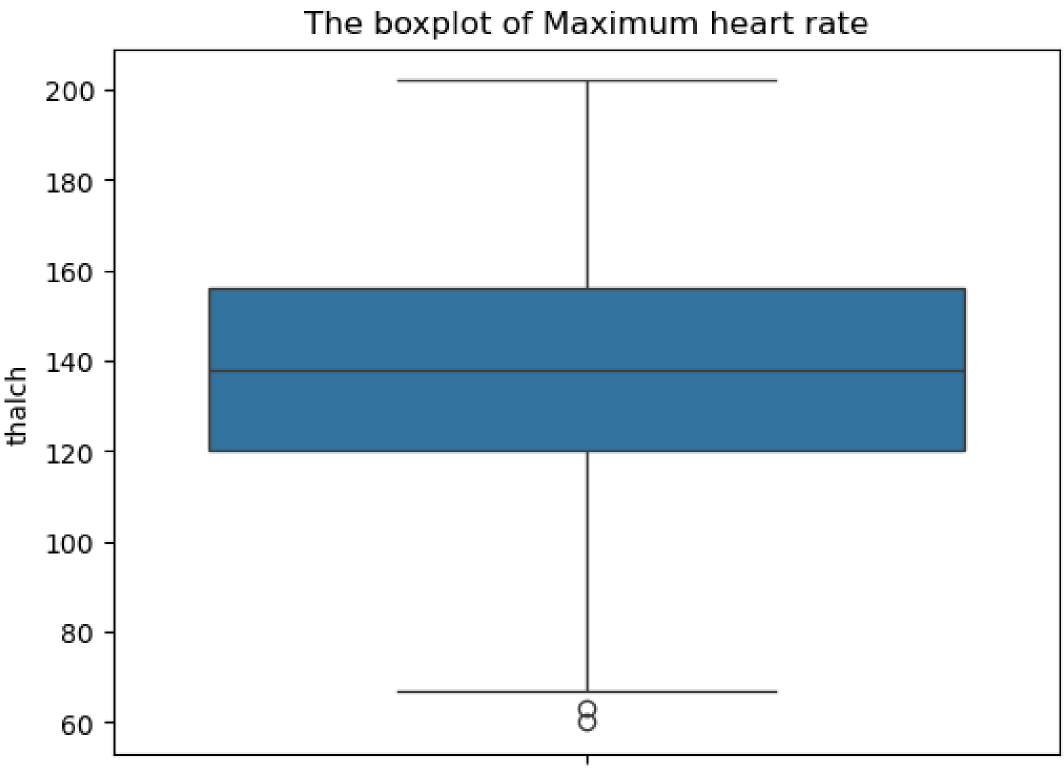
In [46]:

```
Dictionary={"age":"Age" , "trestbps":"Blood Pressure at rest" , "chol":'

for col in df[["age" , "trestbps" ,  "chol" , "thalch"]]:
    sns.boxplot(y=col , data=df)
    plt.xlabel={col}
    plt.title(f"The boxplot of {Dictionary[col]}")
    plt.show()
```









In [47]:
```
from scipy import stats
```

Cholestrol outliers

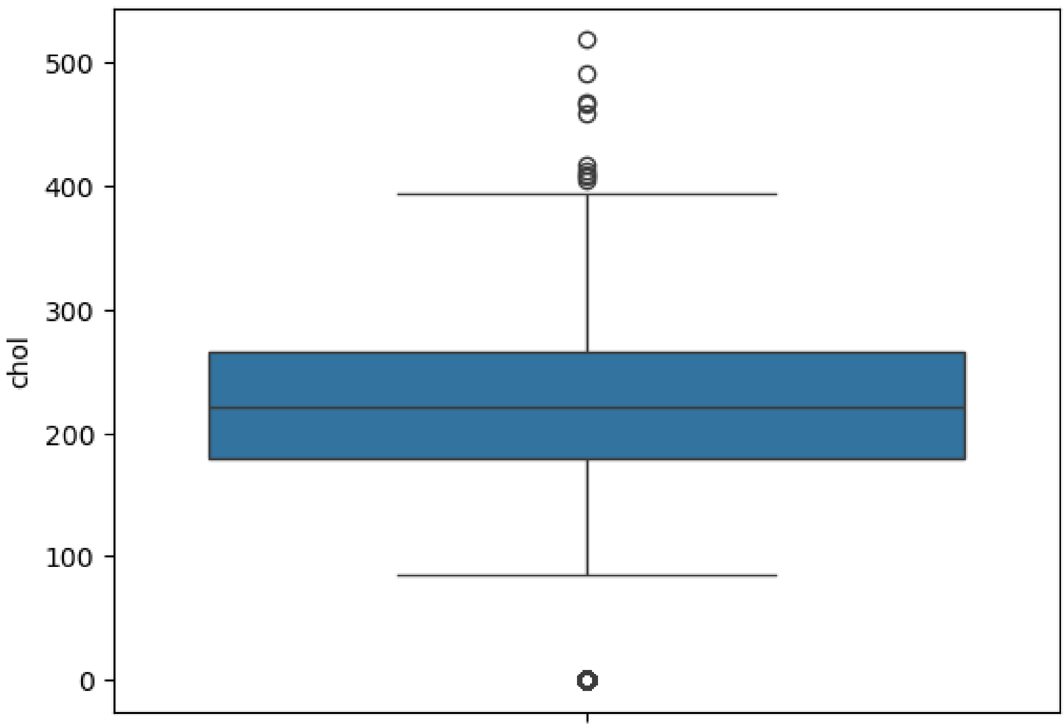
In [48]:
```
# Calculate the z-score for each student's height
z = np.abs(stats.zscore(X['chol']))

# Identify outliers as students with a z-score greater than 3
threshold = 3
outliers = df[z > threshold]

# Print the outliers
print(outliers)
```

```
      id  age  sex  dataset  cp  trestbps   chol  fbs  restecg  thalch  
exang  \
152  153   67    0        0   2     115.0  564.0    0        0   160.0  
0
528  529   32    1        1   0     118.0  529.0    0        1   130.0  
0
546  547   54    1        1   0     130.0  603.0    1        1   125.0  
1

     oldpeak  slope        ca  thal  num
152      1.6      1  0.000000     2    0
528      0.0      1  0.676375     1    1
546      1.0      1  0.676375     1    1
```

In [52]:
```
X.drop([152,153,528,529,546,547],axis=0,inplace=True)
```

In [63]:
```
sns.boxplot(y='chol' , data=X)
```

Out[63]: <Axes: ylabel='chol'>



trestbps outliers

In [49]:

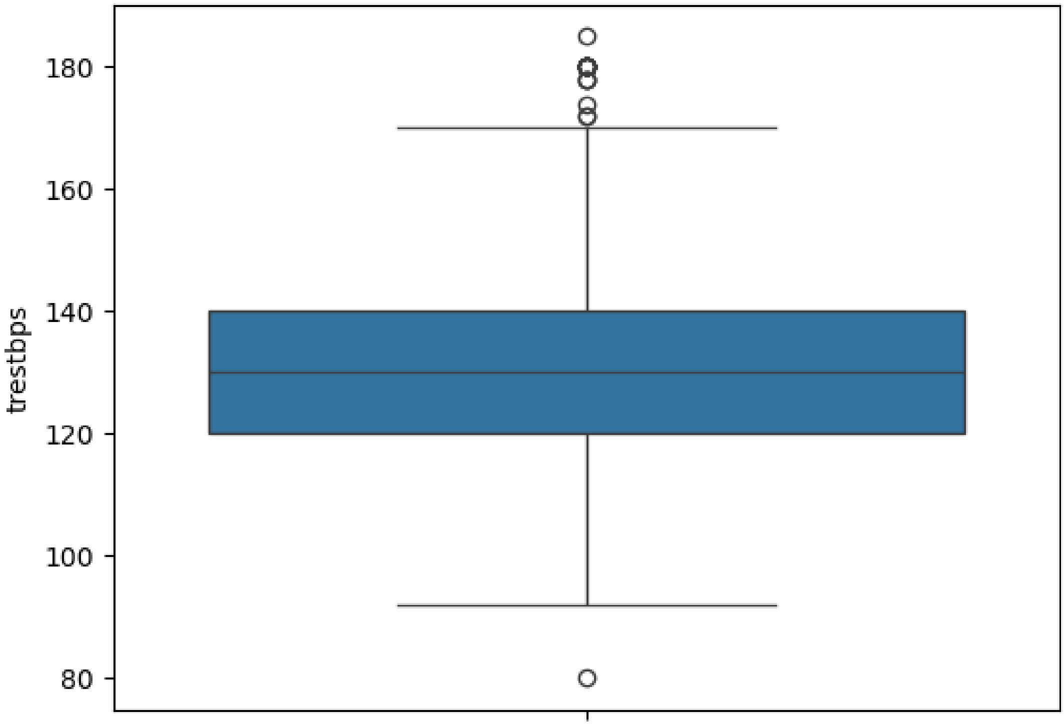
```python
# Calculate the z-score for each student's height
z = np.abs(stats.zscore(X['trestbps']))

# Identify outliers as students with a z-score greater than 3
threshold = 3
outliers = df[z > threshold]

# Print the outliers
print(outliers)
```

```
      id  age  sex  dataset  cp  trestbps   chol  fbs  restecg  thalch
exang  \
126  127   56    0        0   0     200.0  288.0    1        0   133.0
1
188  189   54    1        0   1     192.0  283.0    0        0   195.0
0
338  339   39    1        1   1     190.0  241.0    0        1   106.0
0
548  549   54    1        1   0     200.0  198.0    0        1   142.0
1
680  681   61    1        2   2     200.0    0.0    0        2    70.0
0
701  702   64    0        2   0     200.0    0.0    0        1   140.0
1
753  754   55    1        3   2       0.0    0.0    0        1   155.0
0
896  897   61    1        3   0     190.0  287.0    1        0   150.0
1

     oldpeak  slope        ca  thal  num
126      4.0      0  2.000000     2    3
188      0.0      2  1.000000     2    1
338      0.0      1  0.676375     1    0
548      2.0      1  0.676375     1    1
680      0.0      1  0.676375     1    3
701      1.0      1  0.676375     1    3
753      1.5      1  0.676375     1    3
896      2.0      0  0.676375     1    4
```

In [53]:

```python
X.drop([126,127,188,189,338,339,548,549,680,681,701,702,753,754,896,897]
```

In [61]:
```python
sns.boxplot(y='trestbps' , data=X)
```
Out[61]: <Axes: ylabel='trestbps'>



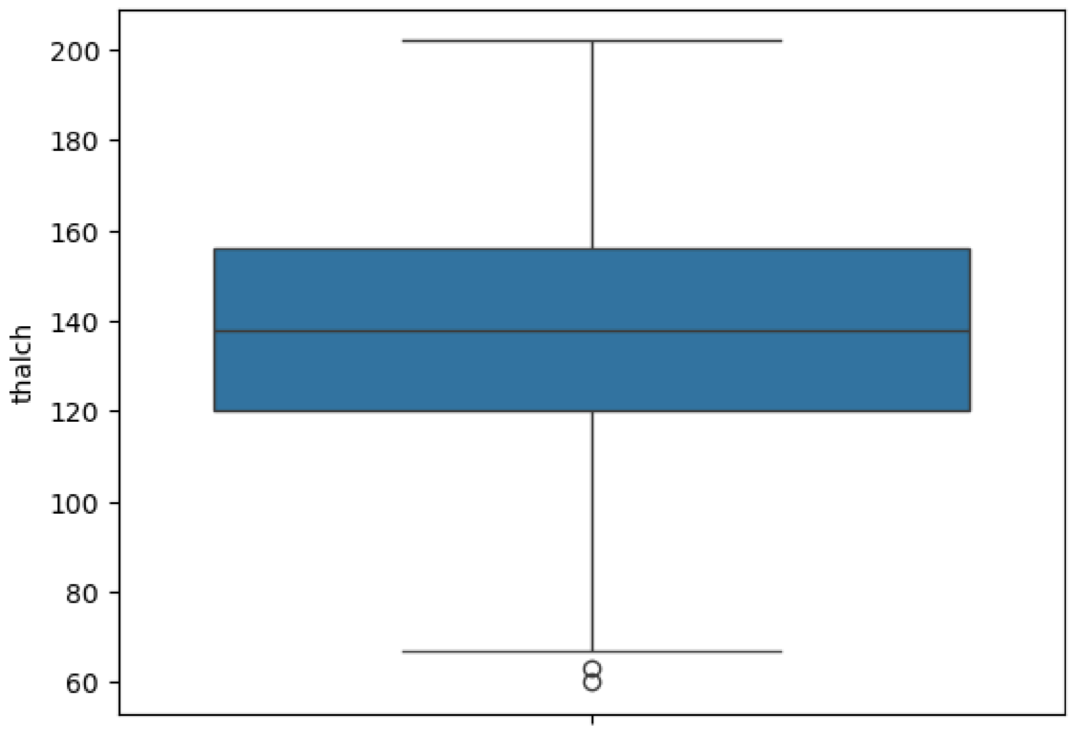
Thalch outliers

In [50]:
```python
# Calculate the z-score for each student's height
z = np.abs(stats.zscore(X['thalch']))

# Identify outliers as students with a z-score greater than 3
threshold = 3
outliers = df[z > threshold]

# Print the outliers
print(outliers)
```
```
      id  age  sex  dataset  cp  trestbps  chol  fbs  restecg  thalch
exang  \
631  632   51    1        2   0     140.0   0.0    0        1    60.0
0

     oldpeak  slope        ca  thal  num
631      0.0      1  0.676375     1    2
```

In [51]:
```python
X.drop([631,632],axis=0,inplace=True)
```

```
In [55]: sns.boxplot(y='thalch' , data=df)
```

Out[55]: <Axes: ylabel='thalch'>




```
In [ ]:
```

Machine Learning Models

```
In [56]: df
```

Out[56]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	o
0	1	63	1	0	3	145.000000	233.0	1	0	150.000000	0	2.
1	2	67	1	0	0	160.000000	286.0	0	0	108.000000	1	1.
2	3	67	1	0	0	120.000000	229.0	0	0	129.000000	1	2.
3	4	37	1	0	2	130.000000	250.0	0	1	187.000000	0	3.
4	5	41	0	0	1	130.000000	204.0	0	0	172.000000	0	1.
...	...	...	...	...	...	...	...	...	...	...	...	
915	916	54	0	3	0	127.000000	333.0	1	2	154.000000	0	0.
916	917	62	1	3	3	132.132404	139.0	0	2	137.545665	0	0.
917	918	55	1	3	0	122.000000	223.0	1	2	100.000000	0	0.
918	919	58	1	3	0	132.132404	385.0	1	0	137.545665	0	0.
919	920	62	1	3	1	120.000000	254.0	0	0	93.000000	1	0.

920 rows × 16 columns

Feature that we will be using in Machine Learning Models building

The Targeted column is num which is the predicted attribute. We will use this column to predict the heart disease. The unique values in this column are: [0,1,2,3,4], which states that there are 5 types of heart diseases.

- 0 = no heart disease.
- 1 = Mild Heart Disease types.
- 2 = Moderate Heart Disease type.
- 3 = Severe Heart Disease type.
- 4 = Critical Heart Disease type.

Standard Scaler

In [57]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)
```

In [ ]:
```
```

In [ ]:
```
```

Enlist all the models that you will use to predict the heart disease. These models should be classifiers for multi_class classification.

1. logistic regression.
2. KNN
3. NB
4. SVM
5. Decision Tree
6. Random Forest
7. XGBoost
8. GradientBoosting
9. AdaBoost
10. lightGBM

IMPORTING ALL MODEL

In [58]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.naive_bayes import GaussianNB

# import metrics
from sklearn.metrics import accuracy_score, confusion_matrix, classifica
```

Create a function for models and evaluate them

In [59]:

```
models = [
    ('Logistic Regression', LogisticRegression(random_state=42)),
    ('Gradient Boosting', GradientBoostingClassifier(random_state=42)),
    ('KNeighbors Classifier', KNeighborsClassifier()),
    ('Decision Tree Classifier', DecisionTreeClassifier(random_state=42)
    ('AdaBoost Classifier', AdaBoostClassifier(random_state=42)),
    ('Random Forest', RandomForestClassifier(random_state=42)),
    ('XGboost Classifier', XGBClassifier(random_state=42)),

    ('Support Vector Machine', SVC(random_state=42)),

    ('Naye base Classifier', GaussianNB()) ]
```

In [60]:

```
results = {}
best_model = None
best_accuracy = 0.0

for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    print(f"Model Accuracy: {name} - {accuracy:.4f}")

    results[name] = accuracy
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_model = name

print(f"\nBest Model: {best_model}")
```

```
Model Accuracy: Logistic Regression - 0.5870
Model Accuracy: Gradient Boosting - 0.6087
Model Accuracy: KNeighbors Classifier - 0.5217
Model Accuracy: Decision Tree Classifier - 0.5924
Model Accuracy: AdaBoost Classifier - 0.5815
Model Accuracy: Random Forest - 0.6033
Model Accuracy: XGboost Classifier - 0.6033
Model Accuracy: Support Vector Machine - 0.5598
Model Accuracy: Naye base Classifier - 0.5109

Best Model: Gradient Boosting
```

In [ ]:

```

```

Outputs:¶

1. The minimum age to have a heart disease start from 28 years old. (by min max age)
2. Most of the people get heart disease at the age of 53 to 54 years. (by age describe)
3. Most of the males and females get are with heart disease at the age of 54 to 55 years.
4. Male percentage inthe data: 78.91%
5. Female percentage in the data : 21.09%
6. Males are 274.23% more than female in the data.
7. We have the highest number of people from Clveland(304) and lowest from Switzerland (123).

..Age vs Sex and origin..

8. The highest number of female in this dataset are from Cleveland(97) and lowest are from VA Long Beach(6).
9. The highest number of male are from Hungary(212) and lowest from Switzerland(113).
   ..Chest pain according to Origins..
10. The high number of Typical angina, Asymptomatic and Non anginal chest pain is in the Cleveland while Atypical anigna is highly occured in Hungary.
11. Lowest number of chest pain (Typical angina, Asymptomatic, Non anginal and Atypical angina)is happened in Switzerland as compare to other origins.

..Chest pain according to Age..

12. TThe highest number of case of chest pain is happened in 'Asymtomatic Angina' is 45 and the lowest number of chest pain is that happened is Typical Angina is 11.

- The highest number of case of 'Typical Angina' occurred among individuals between the ages of 62 and 63. Notably, 6 individuals within this age range were identified as having Typical Angina.
- The highest number of case of 'Asymtomatic Angina' occurred among individuals between the ages of 56 to 57 years. Notably, 47 individuals within this age group were identified as having Asymptomatic Angina.
- The highest number of case of 'Non Anginal' occurred among individuals between the ages of 54 to 55 years. Notably, 19 individuals within this age group were identified as having Non Anginal.
- The highest number of case of 'Atypical Angina' occurred among individuals between the ages of 54 to 55 years. Notably, 28 individuals within this age group were identified as having Atypical Anginal