DATA SCIENCE PROJECT ON HEART DISEASE PREDICTION

BUISENESS CASE : BASE ON GIVEN FEATURE AND MESUREMENT PREDICT WHETHER PATIENT WILL HAVE HEART DISEASE OR NOT

BINARY CLASSIFICATION TASK

IMPORTING THE PYTHON LIBRARIES

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('values.csv')
df1 = pd.read_csv('labels.csv')
data = pd.concat([df,df1],axis=1)
data.head()
```

Out[2]:

	patient_id	slope_of_peak_exercise_st_segment	thal	resting_blood_pre
0	0z64un	1	normal	
1	ryoo3j	2	normal	
2	yt1s1x	1	normal	
3	l2xjde	1	reversible_defect	
4	oyt4ek	3	reversible_defect	

INTRODUCTION OF PROJECT

Cardiovascular disease or heart disease is the leading cause of death amongst women and men and amongst most racial/ethnic groups in the United States. Heart disease describes a range of conditions that affect your heart. Diseases under the heart disease umbrella include blood vessel diseases, such as coronary artery disease. From the CDC, roughly every 1 in 4 deaths each year are due to heart disease. The WHO states that human life style is the main reason behind this heart problem. Apart from this there are many key factors which warns that the person may/may not getting chance of heart disease.

The term heart disease is often used interchangeably with the term cardiovascular disease. Cardiovascular disease generally refers to conditions that involve narrowed or blocked blood vessels that can lead to a heart attack, chest pain (angina) or stroke.

DOMAIN ANALYSIS

TARGET COLUMN == HEART DISEASE PRESENT

- In this project we are going to analyze that how other feature of dataset affecting heart disease

1.PATIENT ID:

Id of particular patient, Id is used to identify a patient,this is a unique column so that its not affect to heart disease

2.SLOPE OF PEAK EXERCISE ST SEGMENT:

While a high ST depression is considered normal & healthy. The “ slope ” hue, refers to the peak exercise ST segment, with values: 1: upsloping, 2: flat, 3: downsloping). Both positive & negative heart disease patients exhibit equal distributions of the 3 slope categories.

3.THAL:

A blood disorder called thalassemia,[normal, reversible defect, fixed defect]

4.RESTING BLOOD PRESSURE:

blood pressure tells a lot about your general health. High blood pressure or hypertension can lead to several heart related issues and other medical conditions.Uncontrolled high blood pressure can lead to stroke.

5.CHEST PAIN TYPE:

Most of the chest pain causes are not dangerous to health, but some are serious, while the least cases are life-threatening.[TA: typical angina(1), ATA: Atypical angina(2), NAP: non-anginal pain(3), ASY: asymtomatic(4)]

6.NUM OF MAJOR VESSELS:

Major Blood Vessels of the Heart: Blood exits the right ventricle through the pulmonary trunk artery. Approximately two inches superior to the base of the heart, this vessel branches into the left and right pulmonary arteries, which transport blood into the lungs.[number of major vessels: 0 to 3]

7.FASTING BLOOD SUGAR:

Your Fasting blood sugar level of 120 is a High Fasting blood sugar level. If your Fasting blood sugar is in between 74 mg/dL and 99 mg/dL, then you need not worry as 74-99 mg/dL is the normal range for Fasting blood sugar. But if your Fasting blood sugar is lesser or greater than the above values, then there may be some problem in your body. (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

8.RESTING EKG/ECG RESULT:

The electrocardiogram (ECG or EKG) is a test that measures the heart's electrical activity, and a resting ECG is administered when the patient is at rest. It involves noninvasive recording with adhesive skin electrodes placed on specially prepared spots on the skin, and it plots out the heart's activity on a graph. It is used to determine the health of the heart and circulatory system and to help diagnose issues with associated body systems.[0: normal, 1:having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), 2:showing probable or definite left ventricular hypertrophy by Estes' criteria]

9.SERUM CHOLESTEROL:

A person's serum cholesterol level represents the amount of total cholesterol in their blood. A person's serum cholesterol level comprises the amount of high-density lipoprotein (HDL), low-density lipoprotein (LDL), and triglycerides in the blood. Triglycerides are a type of fat bundled with cholesterol.

10. oldpeak_eq_st_depression:

oldpeak = ST depression induced by exercise relative to rest, a measure of abnormality in electrocardiograms

11.SEX:

sex (1 = male; 0 = female)

12.AGE:

age in years

13.MAX HEART RATE:

It has been shown that an increase in heart rate by 10 beats per minute was associated with an increase in the risk of cardiac death by at least 20%, and this increase in the risk is similar to the one observed with an increase in systolic blood pressure by 10 mm Hg.[Average heart rate: 60 to 100 bpm]

14.EXERCISE INDUCED ANGINA:

Angina is chest pain or discomfort caused when your heart muscle doesn't get enough oxygen-rich blood.[0: no, 1: yes]

15.HEART DISEASE PRESENT:

[0: no heart disease present, 1: heart disease present]

BASIC CHECKS

In [3]:
```
data.head()
```

Out[3]:

	patient_id	slope_of_peak_exercise_st_segment	thal	resting_blood_pre
0	0z64un	1	normal	
1	ryoo3j	2	normal	
2	yt1s1x	1	normal	
3	l2xjde	1	reversible_defect	
4	oyt4ek	3	reversible_defect	

In [4]:
```
data.tail()
```

Out[4]:

	patient_id	slope_of_peak_exercise_st_segment	thal	resting_blood_p
175	5qfar3	2	reversible_defect	
176	2s2b1f	2	normal	
177	nsd00i	2	reversible_defect	
178	0xw93k	1	normal	
179	2nx10r	1	normal	

In [5]:
```
data.columns
```

Out[5]:
```
Index(['patient_id', 'slope_of_peak_exercise_st_segment', 'thal',
       'resting_blood_pressure', 'chest_pain_type', 'num_major_vessels',
       'fasting_blood_sugar_gt_120_mg_per_dl', 'resting_ekg_results',
       'serum_cholesterol_mg_per_dl', 'oldpeak_eq_st_depression', 'sex',
'age',
       'max_heart_rate_achieved', 'exercise_induced_angina', 'patient_i
d',
       'heart_disease_present'],
      dtype='object')
```

- Patient ID feature return twice

In [6]:
```
data.shape
```

Out[6]:
```
(180, 16)
```

- In data set toatal 180 observations and 15 features

EXAMINE THE DATA

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 16 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   patient_id                            180 non-null    object 
 1   slope_of_peak_exercise_st_segment     180 non-null    int64  
 2   thal                                  180 non-null    object 
 3   resting_blood_pressure                180 non-null    int64  
 4   chest_pain_type                       180 non-null    int64  
 5   num_major_vessels                     180 non-null    int64  
 6   fasting_blood_sugar_gt_120_mg_per_dl  180 non-null    int64  
 7   resting_ekg_results                   180 non-null    int64  
 8   serum_cholesterol_mg_per_dl           180 non-null    int64  
 9   oldpeak_eq_st_depression              180 non-null    float64
 10  sex                                   180 non-null    int64  
 11  age                                   180 non-null    int64  
 12  max_heart_rate_achieved               180 non-null    int64  
 13  exercise_induced_angina               180 non-null    int64  
 14  patient_id                            180 non-null    object 
 15  heart_disease_present                 180 non-null    int64  
dtypes: float64(1), int64(12), object(3)
memory usage: 22.6+ KB
```

FEATURE DATA-TYPES:

- Continuous: oldpeak_eq_st_depression
- Discrete: slope_of_peak_exercise_st_segment,resting_blood_pressure, chest_pain_type,num_major_vessels, fasting_blood_sugar_gt_120_mg_per_dl, resting_ekg_results, serum_cholesterol_mg_per_dl, sex, age, max_heart_rate_achieved, exercise_induced_angina, heart_disease_present
- Object: patient_id, thal

STASTICAL MEASURE OF DATA

In [8]:
```
data.describe()
```

Out[8]:

	slope_of_peak_exercise_st_segment	resting_blood_pressure	chest_pain_type
count	180.000000	180.000000	180.000000
mean	1.550000	131.311111	3.155556
std	0.618838	17.010443	0.938454
min	1.000000	94.000000	1.000000
25%	1.000000	120.000000	3.000000
50%	1.000000	130.000000	3.000000
75%	2.000000	140.000000	4.000000
max	3.000000	180.000000	4.000000

- No constant column avialble in dataset

EXPLOTARY DATA ANALYSIS (EDA)

1.UNIVARIATE ANALYSIS

In [9]:
```
# Changing column name:
data.rename({'slope_of_peak_exercise_st_segment':'sop','resting_blood_pre
             'num_major_vessels':'major_vessels','fasting_blood_sugar_gt_1
             'resting_ekg_results':'ekg_result','serum_cholesterol_mg_per_
             'oldpeak_eq_st_depression':'oldpeak_st_depression','max_heart
              'heart_disease_present':'heart_disease'},inplace=True,axis=1
```

In [10]:
```
univariate = data[['sop','thal','resting_bp','cpt','major_vessels','fasti
                   'serum_cholesterol','oldpeak_st_depression','sex','age'
                    'heart_disease']]
```

In [11]:
```
import sweetviz
sv = sweetviz.analyze(univariate)
sv.show_html()  # Generate default argument
```

```
Done! Use 'show' commands to display/save.   |
                                             | [100%]   00:00 -> (00:00 left)
Report SWEETVIZ_REPORT.html was generated! NOTEBOOK/COLAB USERS: the web b
rowser MAY not pop up, regardless, the report IS saved in your notebook/co
lab files.
```

OBSEVATION:

1.SOP:

1. The slope of peak exercise st segment with 3 unique observation, [1:upsloping, 2:flat, 3:downsloping]
2. 52% patient has upsloping peak of exercise st segment, 42% patient has flat slope of peak exercise st segment and remaning 7% patient has downsloping peak of exercise st segment.

2.THAL:

1. In thal 3 unique categories [normal,reversible defect,fixed defect]
2. In 98 patient normal blood disorder, 74 patient reversible blood disorder and remaing 8 patient fixed blood disorder

3.RESTING BLOOD PRESSURE:

1. The blood pressure range between 94 to 180 with 131.3 average
2. Most of the patient has 130 resting blood pressure

4.CHEST PAIN TYPE:

1. Four type of chest pain [1: typical angina(TA), 2: Atypical angina(ATA), 3: non-anginal pain(NAP), 4: asymtomatic(ASY)]
2. Most number of (46%) patient has asymtomatic chest pain and (32%) patient has non-anginal pain and remaning (16% & 7%) patient has atypical angina nas typical angina chest pain

5.MAJOR VESSELS:

1. The range of major vessels between 0 to 3
2. Most (106)number of major vessels is 0, and remaining major vessels is [37,23,14]

6.FASTING BLOOD SUGAR:

1. Two categories in fasting blood sugar (1 = true, 0 = false)
2. 151 patient fasting blood sugar is less than 120 mg/dl
3. 29 patient fasting blood sugar is greter than 120 mg/dl

7.RESTING EKG/ECG RESULT:

1. In resting ekg/ecg result 3 uqiue value(0,1,2)
2. The most (52%)number of ekg result is 2, as well as 0 (normal) ekg result is 47%

8.SERUM CHOLESTEROL:

1. The range of serum cholesterol between 126 to 564
2. The Most frequent value of serum cholesterol is(239 & 204)

9.OLD PEAK ST DEPRESSION:

1. The range of old peak st depression between 0 to 6.20

10.SEX == [0:Female, 1:male]

1. Most (124) number of patient is male and 56 patient is female

11.AGE:

1. The age range between 29 to 77 year
2. The average age of patient is 54.8 year

12. MAX HEART RATE:

1. The most obesrvation of heart rate between 140 to 180
2. maximum heart rate is 202 and minimum heart rate is 96

13.EXERCISE INDUCED ANGINA: [0: no, 1: yes]

1. 123 patient has no chest pain and reamining 57 patient chest pain

14. HEART DISEASE == TARGET VERIABLE [0:No heart disease, 1:heart disease]

1. 100 patient has no heart disease and 80 patient has heart disease

2.BIVARIATE ANALYSIS

ANALYSIS ON CATEGORICAL VERIABLE WITH RESPECT TO TARGET VERIABLE(HEART DISEASE)

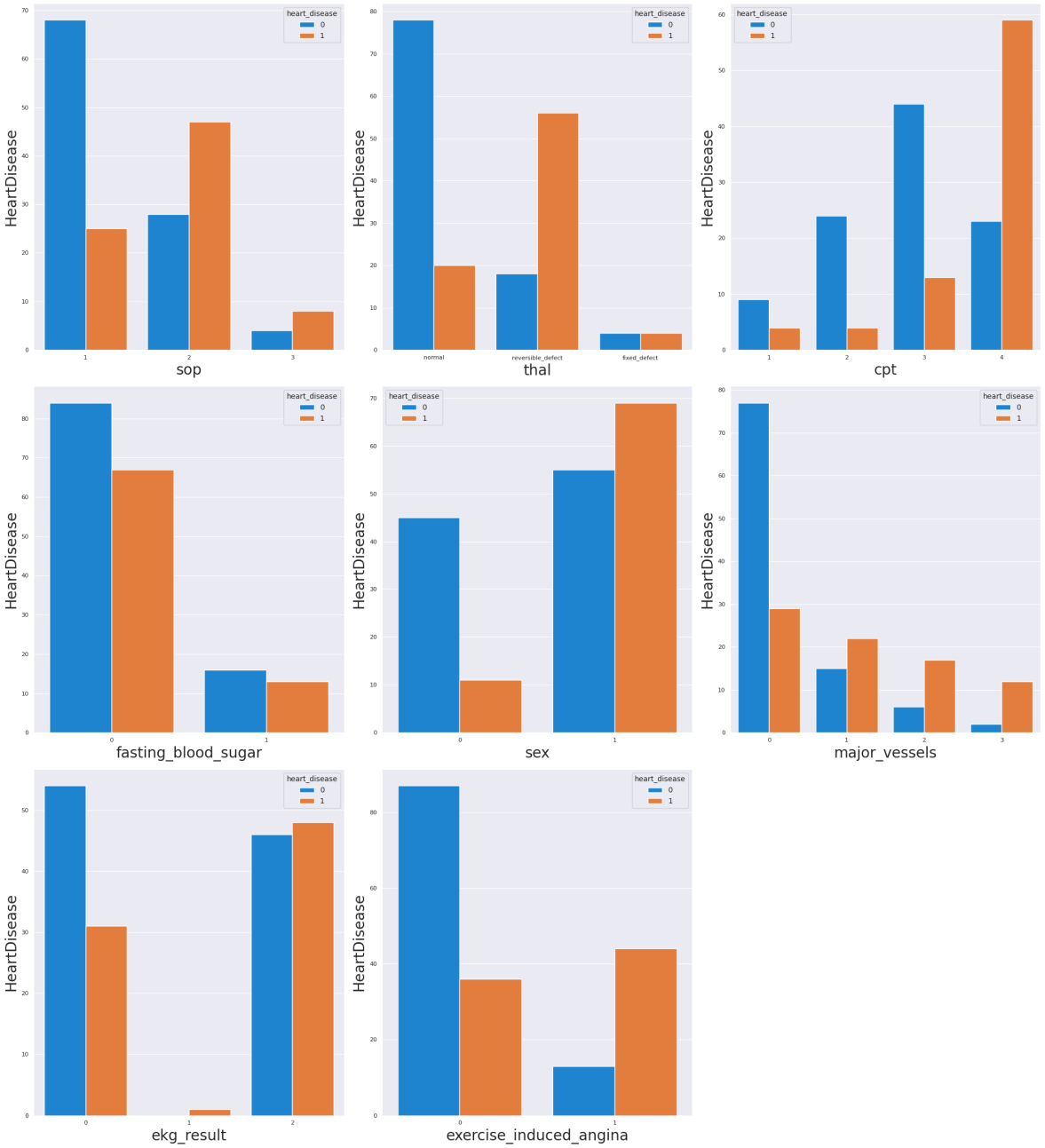
In [12]:
```
data.head()
```

Out[12]:

	patient_id	sop	thal	resting_bp	cpt	major_vessels	fasting_blood_su
0	0z64un	1	normal	128	2	0	
1	ryoo3j	2	normal	110	3	0	
2	yt1s1x	1	normal	125	4	3	
3	l2xjde	1	reversible_defect	152	4	0	
4	oyt4ek	3	reversible_defect	178	1	0	

In [13]:
```
categorical = data[['sop','thal','cpt','fasting_blood_sugar','sex','major
sns.set_style('darkgrid')
plt.figure(figsize=(20,22), facecolor='white')# defining  canvas size
plotnumber = 1 # initializing plotnumber variable to 1 it will maintain t


for column in categorical: # iteration of columns / acessing the columns
    if plotnumber<=9 :     # set the limit
        ax = plt.subplot(3,3,plotnumber)# plotting 9 graphs (3-rows,3-col
        sns.countplot(x=categorical[column],hue=data.heart_disease) # it
        plt.xlabel(column,fontsize=20) # assigning name to x-axis and fon
        plt.ylabel('Heart Attack',fontsize=25) # assigning name to y-axis
    plotnumber+=1 # increment of plotnumber
plt.tight_layout()
plt.show() # used to hide storage area location
```



OBSERVATION:

1.IMPACT OF SOP TO HEART DISEASE:

1. In this table clearly seen the slope of peak st segment is upsloping the chance of heart disease is less than other.
2. If slop pf peak is flat the chance of heart disease in more than upsloping
3. downslope st segment patient is also chance to get heart diseaase
4. with the follwing observation we can say that slope of peak st segment is fullt impact to heart disease

2.IMPACT OF THAL TO HEART DISEASE:

1. Normal blood disorder patient has less chance of heart disease than other thal
2. reversible defect blood disorder has more chance of heart disease and fixed defect blood disorder has 50-50 chance of heart disease

3.IMPACT OF CPT TO HEART DISEASE:

1. if the patient have asymtomatic(4) chest paint the chance of heart disease is more high
2. non-anginal pain(3),typical angina(1), Atypical angina(2) chest pain has less chances of heart disease.
3. but all chest pain types are impacted to heart disease.

4.IMPACT OF FASTING BLOOD SUGAR TO HEART DISEASE:

1. If fasting blood suagar is less than 120mg/dl the chance of heart disease is high.
2. and fasting blood sugar is greter than 120mg/dl the chace of heart disease is slightly less.

5.IMPACT OF SEX TO HEART DISEASE:

Male patient has more chance of heart disease than female

6.IMPACT OF MAJOR VESSELS TO HEART DISEASE:

1. If the major vessels is zero the chance of heart disease is less but zero major vessels are also chance of heart disease
2. 1,2, and 3 major vessels are more(high) chance of heart disease

7.IMPACT OF EKG RESULT TO HEART DISEASE:

1. If the ekg/ecg result is normal(0) the chance of heart disease is less.
2. If ekg/ecg result is 1 the 100% patient has heart disease
3. 2 ekg/ecg result is 50-50% chance of heart disease

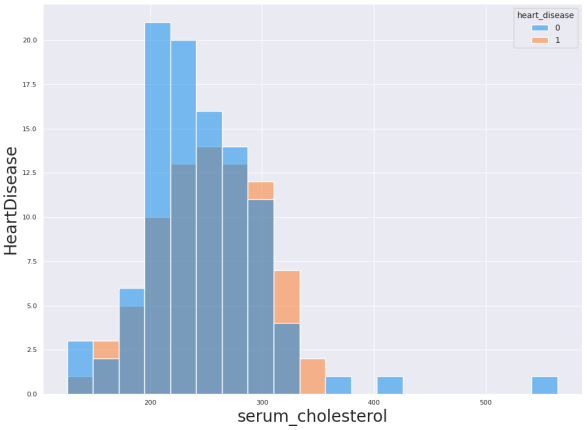
8.IMPACT OF EXERCISE INDUCED ANGINA:

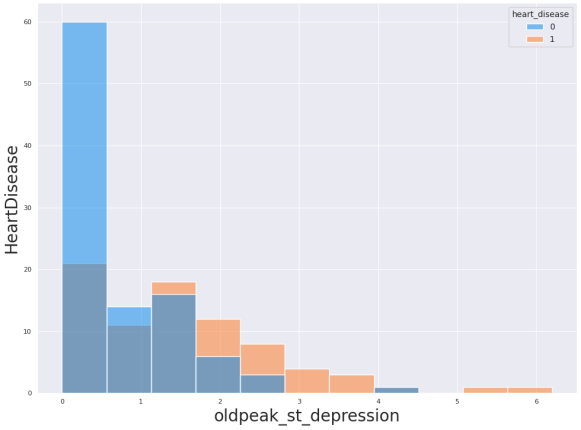
1. If the patient has no chest pain the chance of heart disease is less
2. If patient has chest pain the chance of heart disease is more

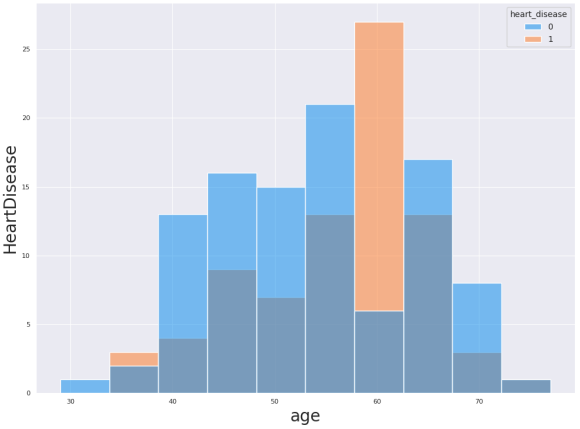
ANALYSIS ON NUMERICAL VERIABLE WITH RESPECT TO TARGET VERIBLE (HEART DISEASE)

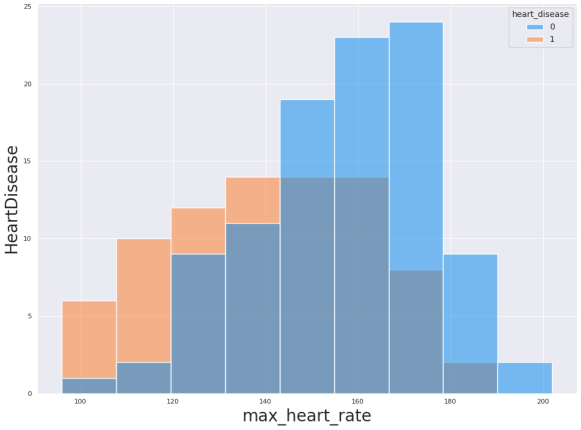
In [14]:
```
numerical = data[['resting_bp','serum_cholesterol','oldpeak_st_depression
sns.set_style('darkgrid')
plt.figure(figsize=(20,22)) # defining canvas size
plotno = 1 # counter

for column in numerical: # iteration of columns / acessing the columns fr
    if plotno<=9:    # set the limit
        plt.subplot(3,2,plotno) # # plotting 5 graphs (3-rows,3-columns)
        sns.histplot(x=numerical[column],hue=data.heart_disease) # it giv
        plt.ylabel('HeartDisease',fontsize=20)# assigning name to y-axis
        plt.xlabel(column,fontsize=20)  # assigning name to x-axis and fo
    plotno+=1 # counter increment
plt.tight_layout()
plt.show() # used to hide the storage loction
```









OBSERVATION:

1.IMPACT OF RESTING BP TO HEART DISEASE:

1. If the blood pressure range between 110 to 150 the chance of heart disease is more
2. If resting blood pressure is low the chance of heart disease is slightly less

2.IMPACT OF SERUM CHOLESTREOL TO HEART DISEASE:

1. If serum cholestreol is less than 350 the heart disease chance is 50-50 percent.
2. serum cholestreol is more than 350 thier is no chance of heart disease

3.IMPACT OF OLD PEAK DEPRESSION TO HEART DISEASE:

1. If old peak depression is less the chance of heart disease is less
2. old peak depression is more than 1 the chance of heart disease is more

4.IMAPCT OF AGE TO HEART DISEASE:

1. At the age of 60 the more chance of heart disease and age range between 40 to 70 heart disease chance is 50-50 percent
2. If age is less than 30 their is no chance of heart disease

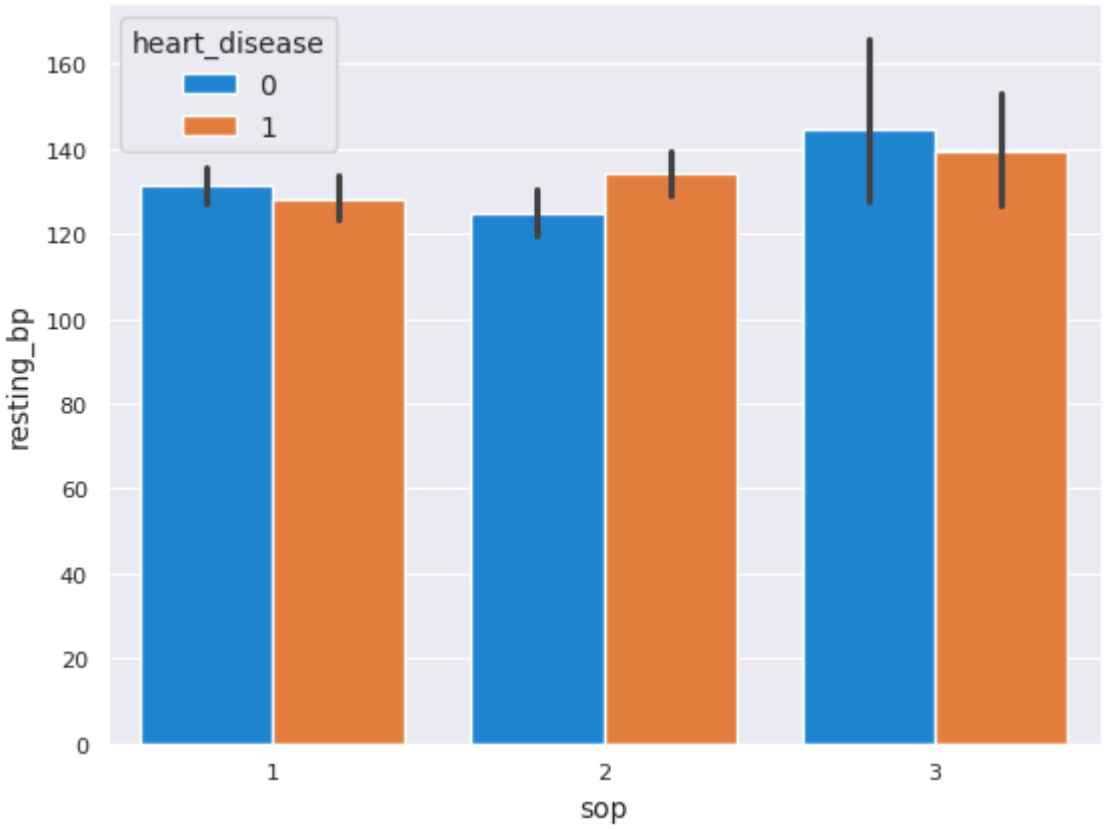
5.IMAPCT OF MAX HEART RATE TO HEART DISEASE:

1. If the heart rate is less than 140 the chance of heart disease is more
2. Above 140 heart rate chance of heart disease is 50-50 percent.
3. If the heart rate is more than 180 their is no chance of heart disease

3.MULTIVARIATE ANALYSIS

CHECK RELATION OF TWO VERIABLE WITH RESPECT TO TARGET VERIBLE (HEART DISEASE)

In [15]:
```
sns.barplot(x='sop',y='resting_bp',hue=data.heart_disease,data=data)
plt.show()
```

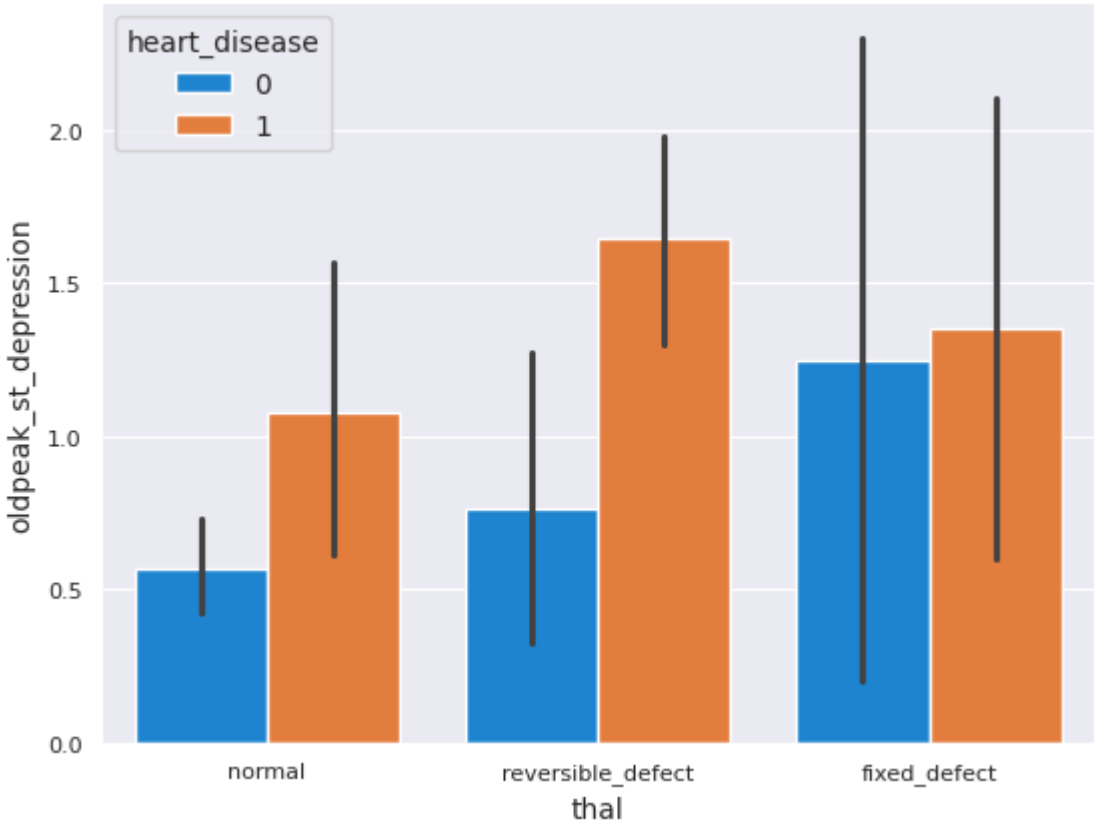


OBSERVATION:

RELATION BETWEEN SOP AND RESTING BLOOD PRESSURE WITH RESPECT TO TARGET VERIABLE(HEART DISEASE)

- In this plot clearly seen that if the resting blood presure is increases the chance of heart disease is equal

In [16]:
```
sns.barplot(x='thal',y='oldpeak_st_depression',hue=data.heart_disease,dat
plt.show()
```

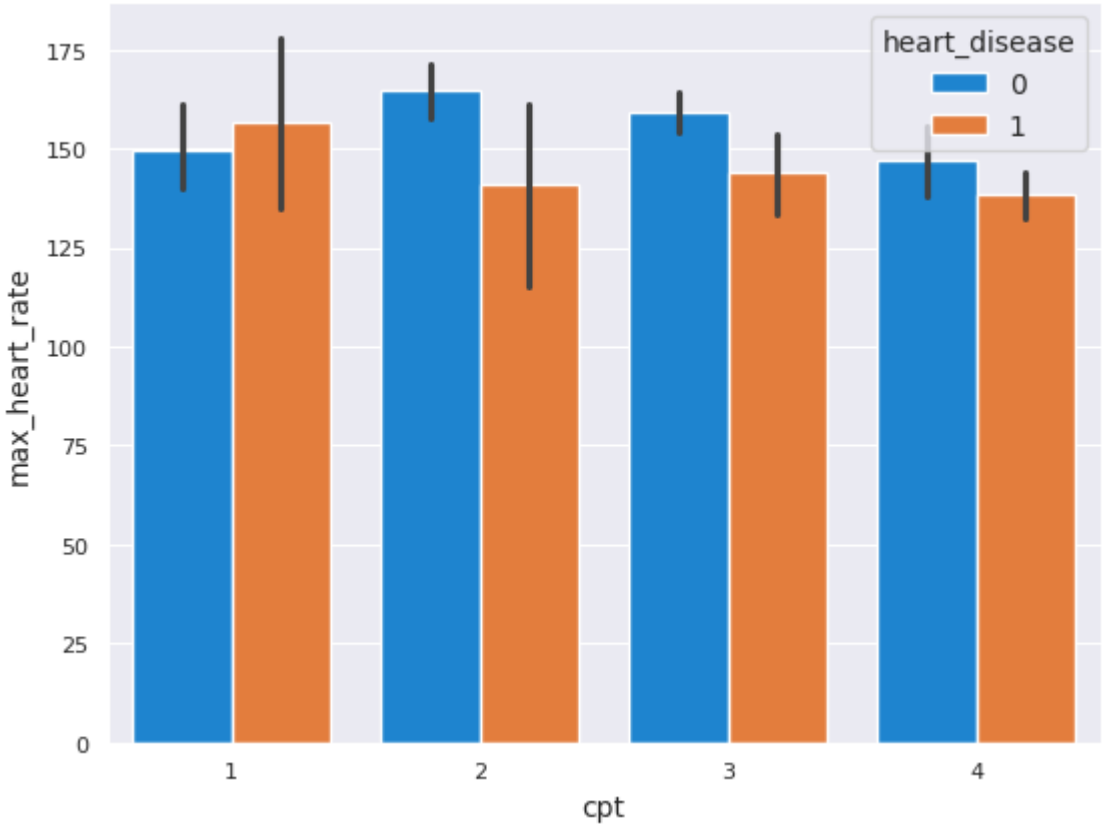


OBSERVATION:

RELATION BETWEEN THAL AND OLD PEAK ST SEGMENT WITH RESPECT TO TARGET VERIABLE(HEART DISEASE)

1. reversible defect blood disorder and high oldpeak st depression the chance of heart disease is more
2. fixed defect blood disorder and above 1 oldpeak st depression then the chance of heart disease is equal
3. If thal is normal and old peak st depression is more than 1 the chance of heart disease is more

In [17]:
```
sns.barplot(x='cpt',y='max_heart_rate',hue=data.heart_disease,data=data)
plt.show()
```

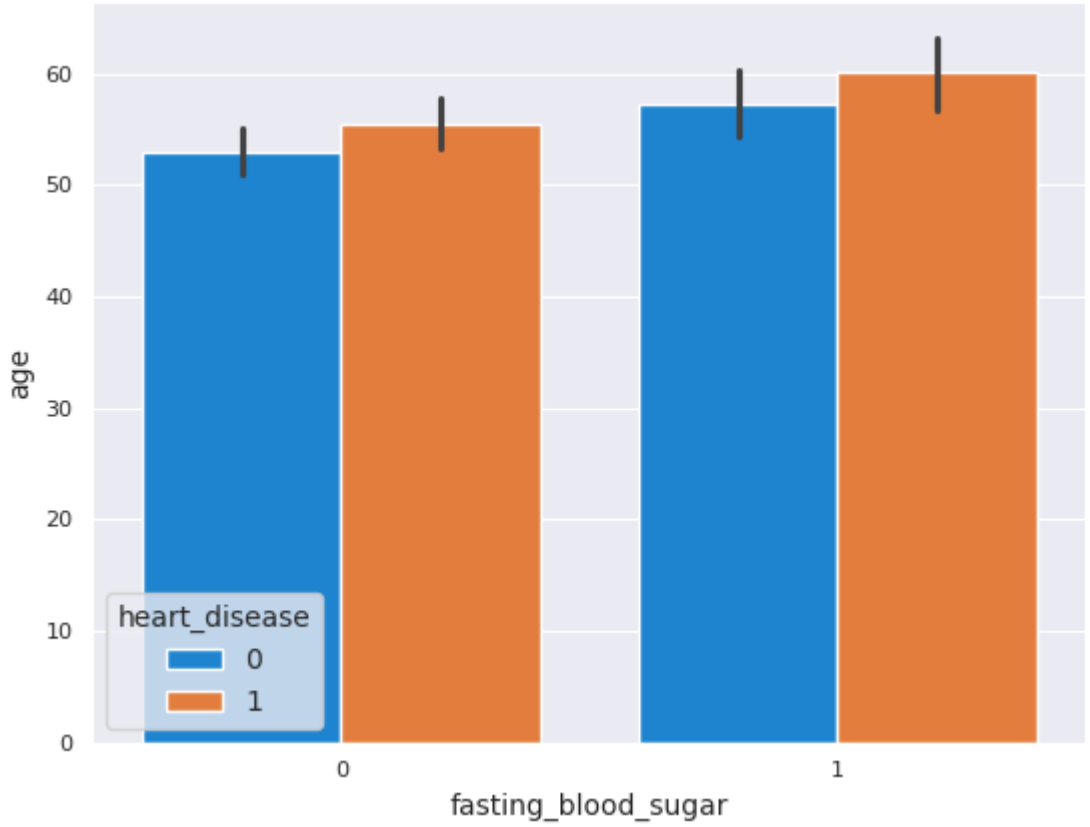


OBSERVATION:

RELATION BETWEEN CPT AND MAX HEART RATE WITH RESPECT TO TARGET VERIABLE(HEART DISEASE)

- In this plot cleary seen that the all types of chest pain and below 150 heart rate are chnace of heart disease is equal

In [18]:
```
sns.barplot(x='fasting_blood_sugar',y='age',hue=data.heart_disease,data=d
plt.show()
```

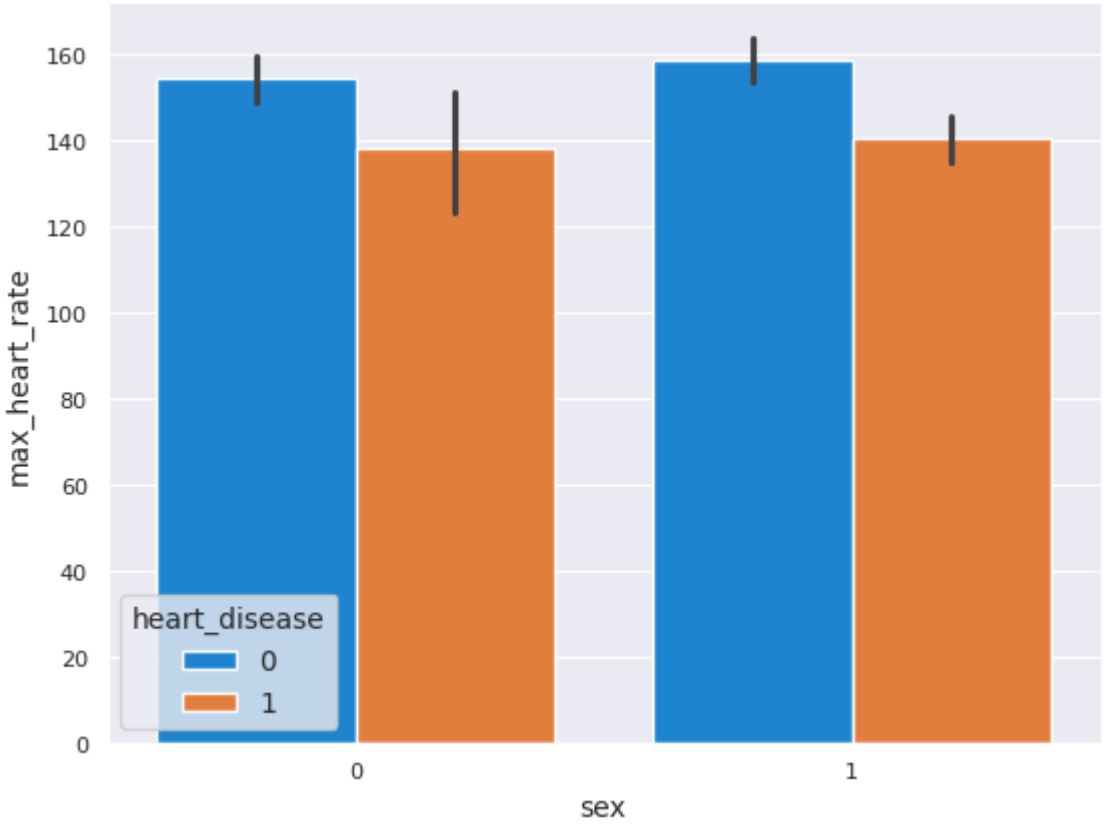


OBSERVATION:

RELATION BETWEEN FASTING BLOOD SUGAR AND AGE WITH RESPECT TO TARGET VERIABLE HEART DISEASE

- If the fasting blood sugar is less than 120mg/dl & greter than 120mg/dl with age 0 to 60 the chance of heart disease is 50-50 percent

In [19]:
```
sns.barplot(x='sex',y='max_heart_rate',hue=data.heart_disease,data=data)
plt.show()
```

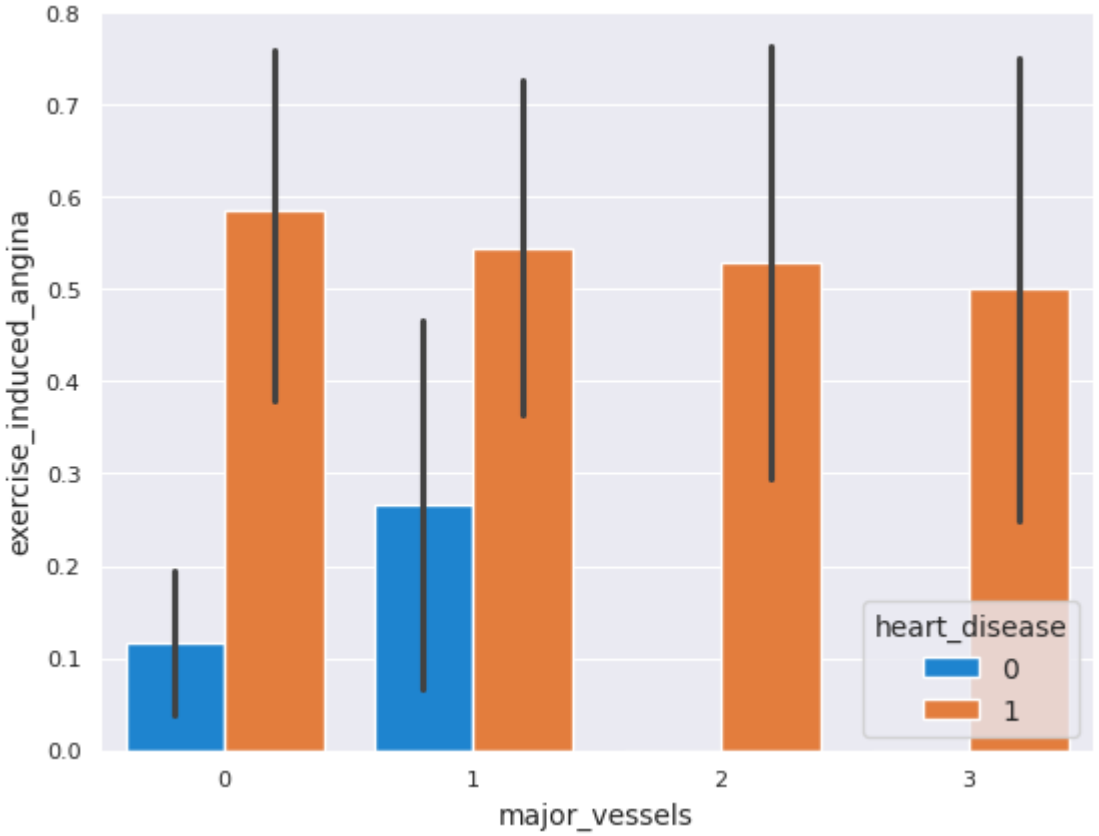


OBSERVATION:

RELATION BETWEEN SEX AND MAX HEART RATE WITH RESPECT TO TARGET VERIBALE(HEART DISEASE)

1. Maximum heart rate is less chance of heart disease in male and female.
2. Minimum heart rate is more chance of heart disease in male and female.

In [20]:
```
sns.barplot(x='major_vessels',y='exercise_induced_angina',hue=data.heart_
plt.show()
```

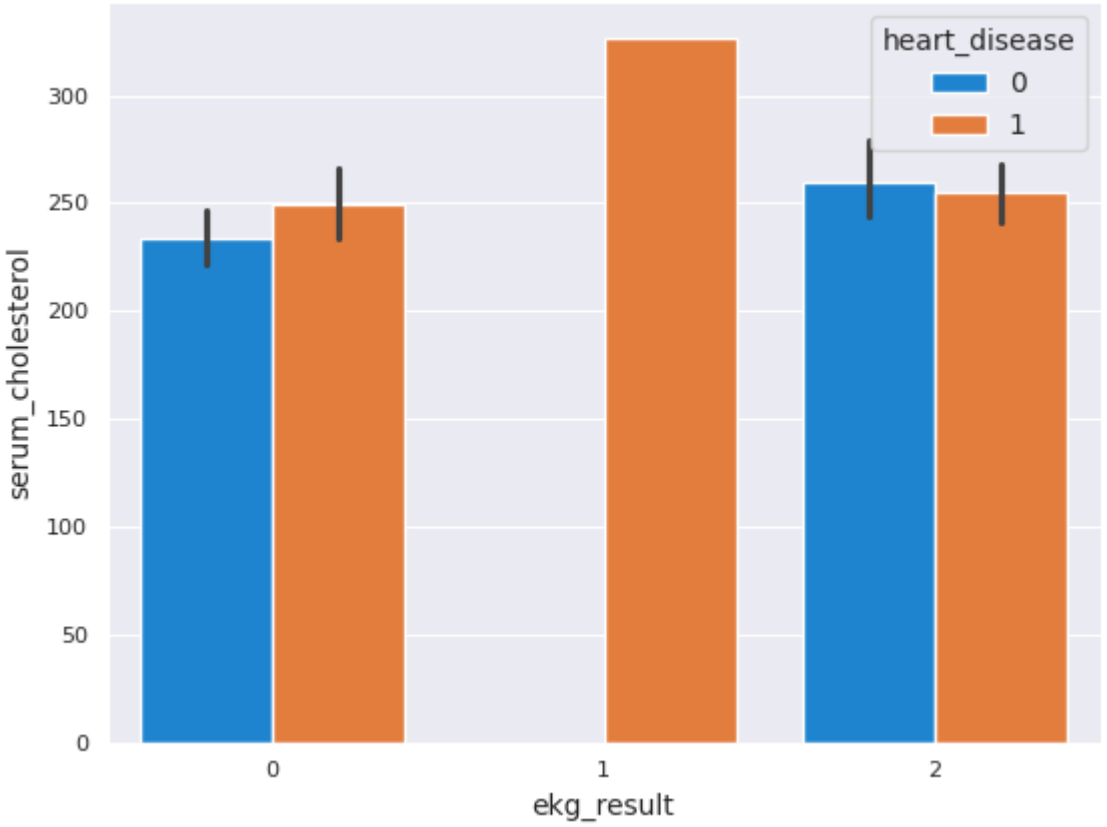


OBSERVATION:

RELATION BETWEEN MAJOR VESSELS AND EXCERCISE INDUCED ANGINA WITH RESPECT TO TARGET VERIBALE(HEART DISEASE)

1. If patient exercise induced angina is minimumn the chance of heart disease in less in 0 and 1 major vessels.
2. Maximum exercise induced angina is more chances of heart disease with all major vassels(0,1,2,3)

In [21]:
```
sns.barplot(x='ekg_result',y='serum_cholesterol',hue=data.heart_disease,d
plt.show()
```



OBSERVATION:

RELATION BETWEEN EKG RESULT AND SERUM CHOLESTEROL WITH RESPECT TO TARGET VERIBALE(HEART DISEASE)

1. 1 ekg/ecg result with all serum cholesterol the 100% chance of heart disease
2. If ekg/ecg result is (0,2) with all serum cholesterol the chance of heart disease is equal.

DATA PREPROCESSING / FEATURE ENGINEERING

1.CHECK MISSING VALUE

In [22]:
```
data.isnull().sum()
```

Out[22]:
```
patient_id                0
sop                       0
thal                      0
resting_bp                0
cpt                       0
major_vessels             0
fasting_blood_sugar       0
ekg_result                0
serum_cholesterol         0
oldpeak_st_depression     0
sex                       0
age                       0
max_heart_rate            0
exercise_induced_angina   0
patient_id                0
heart_disease             0
dtype: int64
```

- Their is no missing value in data

2.CATEGORICAL DATA CONVERSION

MANUAL ENCODING

- Manual encoding is best technique to handle categorical data with the help of map function

In [23]:
```
# In this dataset only one categorical data-type feature
# we use manual encoding to convert categorical data to numerical
# Getting the value counts of thal
data.thal.value_counts()
```

Out[23]:
```
thal
normal               98
reversible_defect    74
fixed_defect          8
Name: count, dtype: int64
```

In [24]:
```
data.thal = data.thal.map({'normal':2,'reversible_defect':1,'fixed_defect
# normal is assign with value 2 because normal has more weightage
# reversible defect assign with value 1 because of less weightage than no
# fixed defect assign with value 0 because of less weightage
```

In [25]:
```
# checking the unique value of thal
data.thal.unique()
```

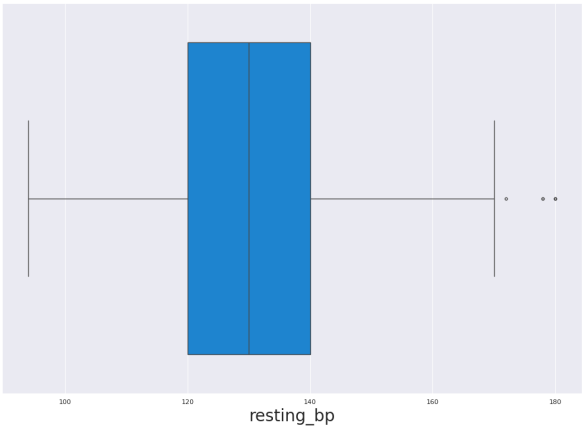
Out[25]:
```
array([2, 1, 0])
```

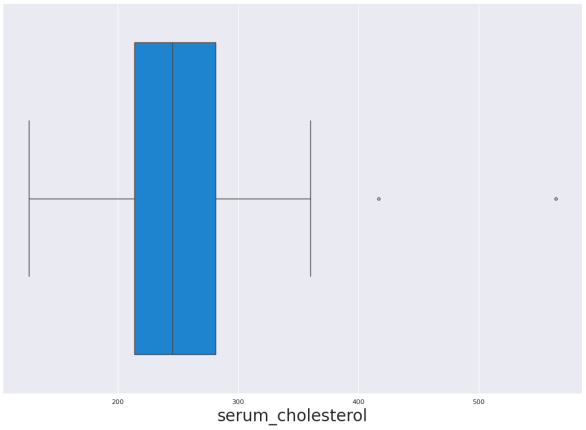
3.OUTLIER HANDLING

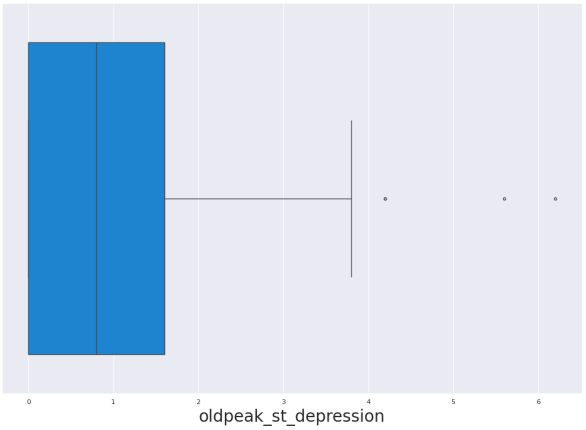
- Handling outlier if it is necessary

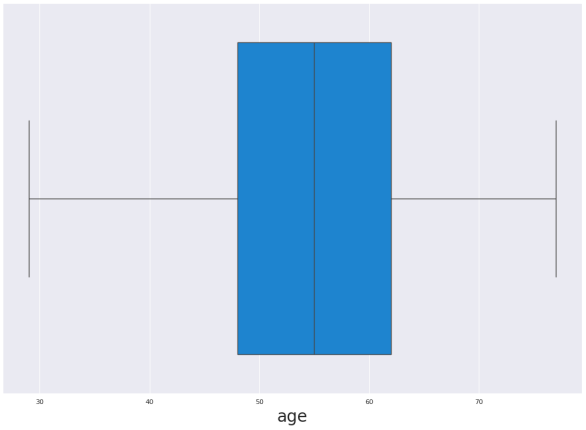
In [26]:
```
out = data[['resting_bp','serum_cholesterol','oldpeak_st_depression','age
sns.set_style('darkgrid')
plt.figure(figsize=(20,22)) # defining canvas size
plotno = 1 # counter
```

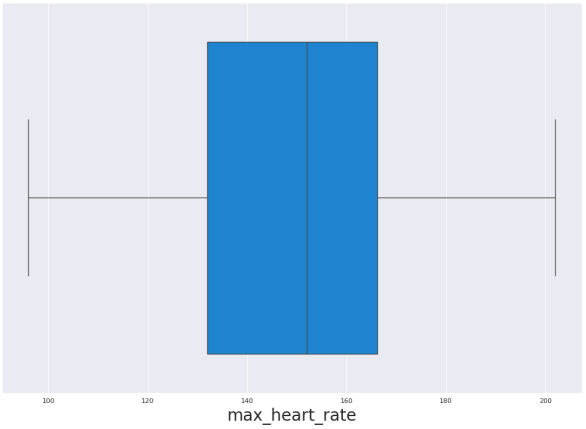
```
for column in out: # iteration of columns / acessing the columns from  da
    if plotno<=6:    # set the limit
        plt.subplot(3,2,plotno) # # plotting 5 graphs (3-rows,2-columns)
        sns.boxplot(x=out[column]) # Plotting box plots
        plt.xlabel(column,fontsize=20)  # assigning name to x-axis and fo
    plotno+=1 # counter increment
plt.tight_layout()
plt.show() # used to hide the storage loction
```









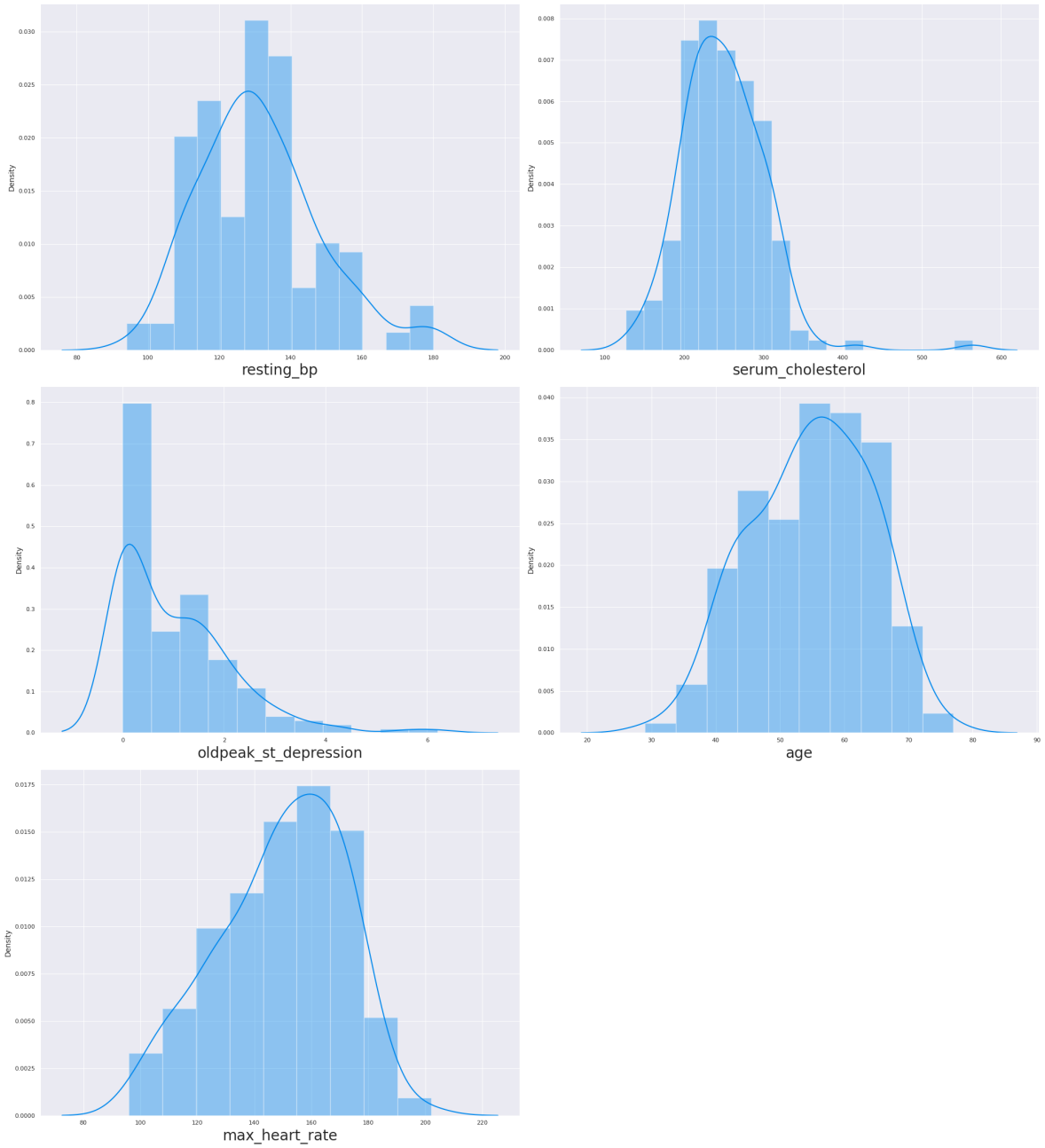


CHECKING DISTRIBUTION BEFORE HANDLE OUTLIER

In [27]:
```
out = data[['resting_bp','serum_cholesterol','oldpeak_st_depression','age
sns.set_style('darkgrid')
plt.figure(figsize=(20,22)) # defining canvas size
plotno = 1 # counter

for column in out: # iteration of columns / acessing the columns from  da
    if plotno<=6:    # set the limit
        plt.subplot(3,2,plotno) # # plotting 5 graphs (3-rows,2-columns)
```

```
        sns.distplot(x=out[column]) # Plotting dist plots
        plt.xlabel(column,fontsize=20)  # assigning name to x-axis and fo
    plotno+=1 # counter increment
plt.tight_layout()
plt.show() # used to hide the storage loction
```



1.RESTING BP

IQR

In [28]:
```
# Use iqr because of some right skewed in data

# Step:1
from scipy import stats
iqr = stats.iqr(data['resting_bp'],interpolation='midpoint')
print("IQR",iqr)

# step:2
Q1 = data['resting_bp'].quantile(0.25)  # first quantile
Q3 = data['resting_bp'].quantile(0.75)  #third quantile
# getting max & min limit
```

```
min_limit = Q1 - 1.5*iqr
print('minimum limit',min_limit)
max_limit = Q3 + 1.5*iqr
print('maximum limit',max_limit)
```

```
IQR 20.0
minimum limit 90.0
maximum limit 170.0
```

In [29]:
```
# Step:3 Filtering the data
data.loc[data['resting_bp'] < min_limit]
```

Out[29]:

patient_id	sop	thal	resting_bp	cpt	major_vessels	fasting_blood_sugar	ekg_re

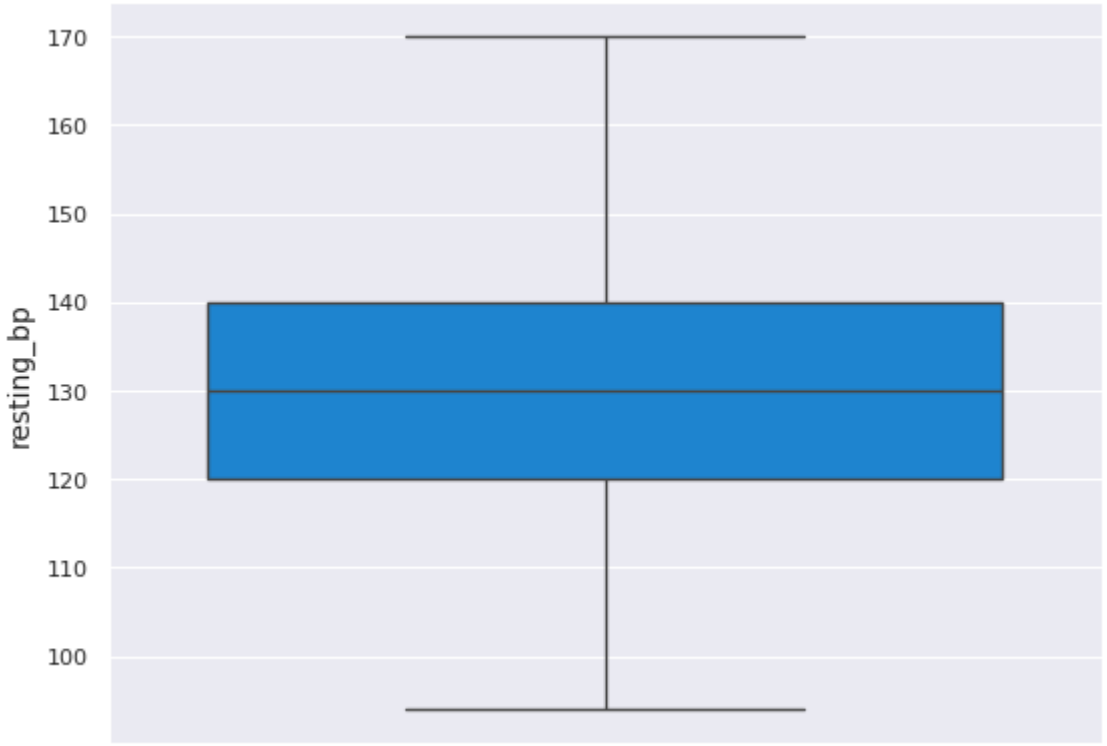
In [30]:
```
data.loc[data['resting_bp'] > max_limit]
```

Out[30]:

	patient_id	sop	thal	resting_bp	cpt	major_vessels	fasting_blood_sugar	ekg
4	oyt4ek	3	1	178	1	0	0	
33	0n5fu0	1	2	180	4	0	0	
72	qwj1yf	1	1	172	3	0	1	
75	4v0q7o	2	1	178	4	2	1	
113	sqddbc	2	1	180	3	0	1	
176	2s2b1f	2	2	180	4	0	0	

In [31]:
```
# Step:4 Imputation of outlier
data.loc[data['resting_bp'] > max_limit ,'resting_bp']=np.median(data['re
```

In [32]:
```
# Step:5 Visualise outlier after imputation
sns.boxplot(data.resting_bp)
plt.show()
```



- After imputation no outlier present in data

2.SERUM CHOLESTEROL

IQR

In [33]:
```
# iqr used because of right skewed in data

# Step:1
iqr = stats.iqr(data['serum_cholesterol'], interpolation='midpoint')
print("IQR",iqr)

# Step:2
Q1 = data['serum_cholesterol'].quantile(0.25)  # first quantile
Q2 = data['serum_cholesterol'].quantile(0.75)  # Third quantile

# Getting maximum and minimum limit
min_limit = Q1 - 1.5*iqr
print('Minimum limit',min_limit)

max_limit = Q3 + 1.5*iqr
print('Maximum limit',max_limit)
```

```
IQR 68.0
Minimum limit 111.75
Maximum limit 242.0
```

In [34]:
```
# Step:3 filtering the data
data.loc[data['serum_cholesterol'] < min_limit]
```

Out[34]:

patient_id	sop	thal	resting_bp	cpt	major_vessels	fasting_blood_sugar	ekg_re

In [35]:
```
data.loc[data['serum_cholesterol'] > max_limit]
len(data.loc[data['serum_cholesterol'] > max_limit])/180*100
```

Out[35]: 53.333333333333336

- In serum cholesterol feature outlier is more than 5 percent so we not need to handle outlier

3.OLD PEAK DEPRESSION

IQR

In [36]:
```
# iqr used becused of right skewed in data

# Step:1
iqr = stats.iqr(data['oldpeak_st_depression'] ,interpolation='midpoint')
print("IQR",iqr)

# Step:2
Q1 = data['oldpeak_st_depression'].quantile(0.25) # first quantile
Q3 = data['oldpeak_st_depression'].quantile(0.75) # Third quantile

# Getting maximum and minimum kimit
min_limit = Q1 - 1.5*iqr
print("Minimum limit",min_limit)

max_limit = Q3 + 1.5*iqr
print("Maximum limit",max_limit)
```

```
IQR 1.6
Minimum limit -2.4000000000000004
Maximum limit 4.0
```

In [37]:
```
# Step:3 filtering the data
data.loc[data['oldpeak_st_depression'] < min_limit]
```

Out[37]:

patient_id	sop	thal	resting_bp	cpt	major_vessels	fasting_blood_sugar	ekg_res

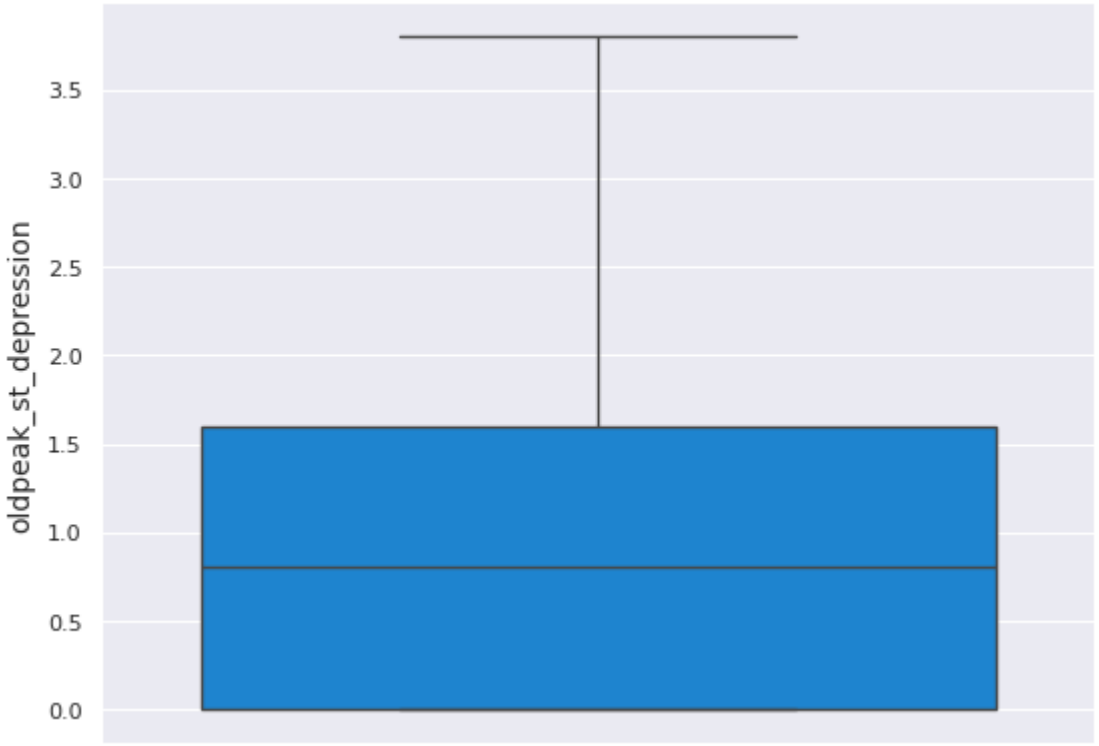
In [38]:
```
data.loc[data['oldpeak_st_depression'] > max_limit]
```

Out[38]:

	patient_id	sop	thal	resting_bp	cpt	major_vessels	fasting_blood_sugar	ekg
4	oyt4ek	3	1	130	1	0	0	
112	6r9x2j	2	1	140	4	3	0	
140	noxsnw	3	1	140	4	0	0	
162	usnkhx	3	1	160	4	3	0	

In [39]:
```
# Step:4 Impute outlier
data.loc[data['oldpeak_st_depression'] > max_limit,'oldpeak_st_depression
```

In [40]:
```
# Step:5 Visualise outlier after imputation
sns.boxplot(data.oldpeak_st_depression)
plt.show()
```

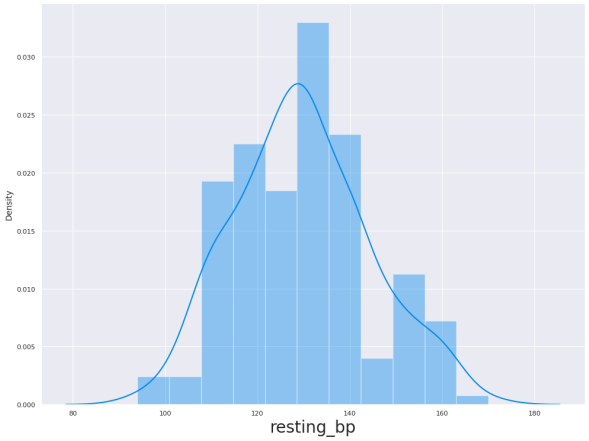


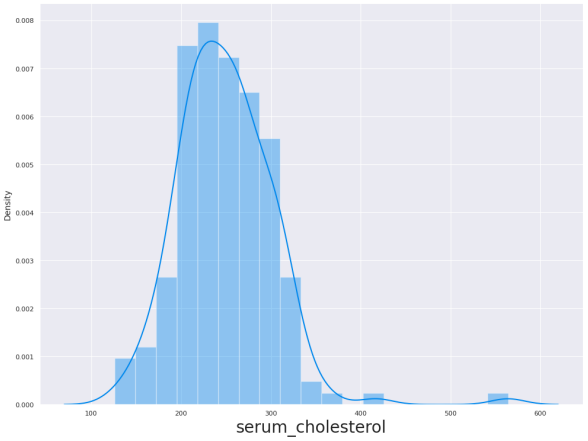
- After imputation no outlier present in data

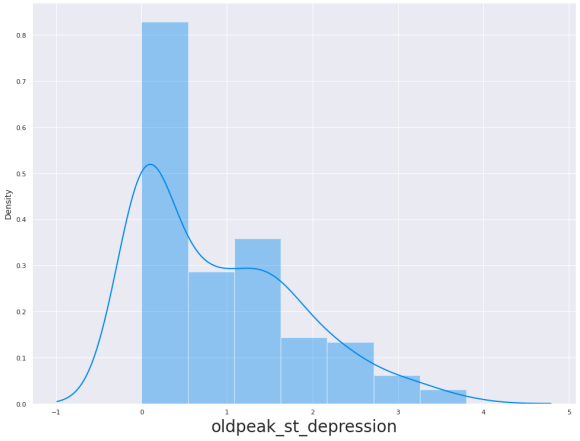
CHECKING DISTRIBUTION AFTER HANDLE OUTLIER

In [41]:
```
out = data[['resting_bp','serum_cholesterol','oldpeak_st_depression']]
sns.set_style('darkgrid')
plt.figure(figsize=(20,22)) # defining canvas size
plotno = 1 # counter

for column in out: # iteration of columns / acessing the columns from  da
    if plotno<=6:    # set the limit
        plt.subplot(3,2,plotno) # # plotting 3 graphs (3-rows,2-columns)
        sns.distplot(x=out[column]) # Plotting dist plots
        plt.xlabel(column,fontsize=20)  # assigning name to x-axis and fo
    plotno+=1 # counter increment
plt.tight_layout()
plt.show() # used to hide the storage loction
```







4.FEATURE SCALING

STANDARD SCALING

- Standard scaling is used because of features are mesure in different units as well as some feature are followed normal distribution
- Standard scaling range : -3 to +3

In [42]:
```
# import library
from sklearn.preprocessing import StandardScaler

# Creating object
scale = StandardScaler()

# Scaling the feature
data[['resting_bp','serum_cholesterol','oldpeak_st_depression','age','max
```

FEATURE SELECTION

1.DROP UNIQUE AND CONSTANT COULMN

In [43]:
```
# Check constant column
data.describe()
```

Out[43]:

	sop	thal	resting_bp	cpt	major_vessels	fasting_bl
count	180.000000	180.000000	1.800000e+02	180.000000	180.000000	1
mean	1.550000	1.500000	1.578984e-16	3.155556	0.694444	
std	0.618838	0.583765	1.002789e+00	0.938454	0.969347	
min	1.000000	0.000000	-2.450820e+00	1.000000	0.000000	
25%	1.000000	1.000000	-6.664645e-01	3.000000	0.000000	
50%	1.000000	2.000000	1.982617e-02	3.000000	0.000000	
75%	2.000000	2.000000	7.061168e-01	4.000000	1.000000	
max	3.000000	2.000000	2.764989e+00	4.000000	3.000000	

- No costant column avilable in dataset

In [44]:
```
# dropping the unique column
data.drop('patient_id',axis=1,inplace=True)
data.head()
```

Out[44]:

	sop	thal	resting_bp	cpt	major_vessels	fasting_blood_sugar	ekg_result	serun
0	1	2	-0.117432	2	0	0	2	
1	2	2	-1.352755	3	0	0	0	
2	1	2	-0.323319	4	3	0	2	
3	1	1	1.529666	4	0	0	0	
4	3	1	0.019826	1	0	0	2	

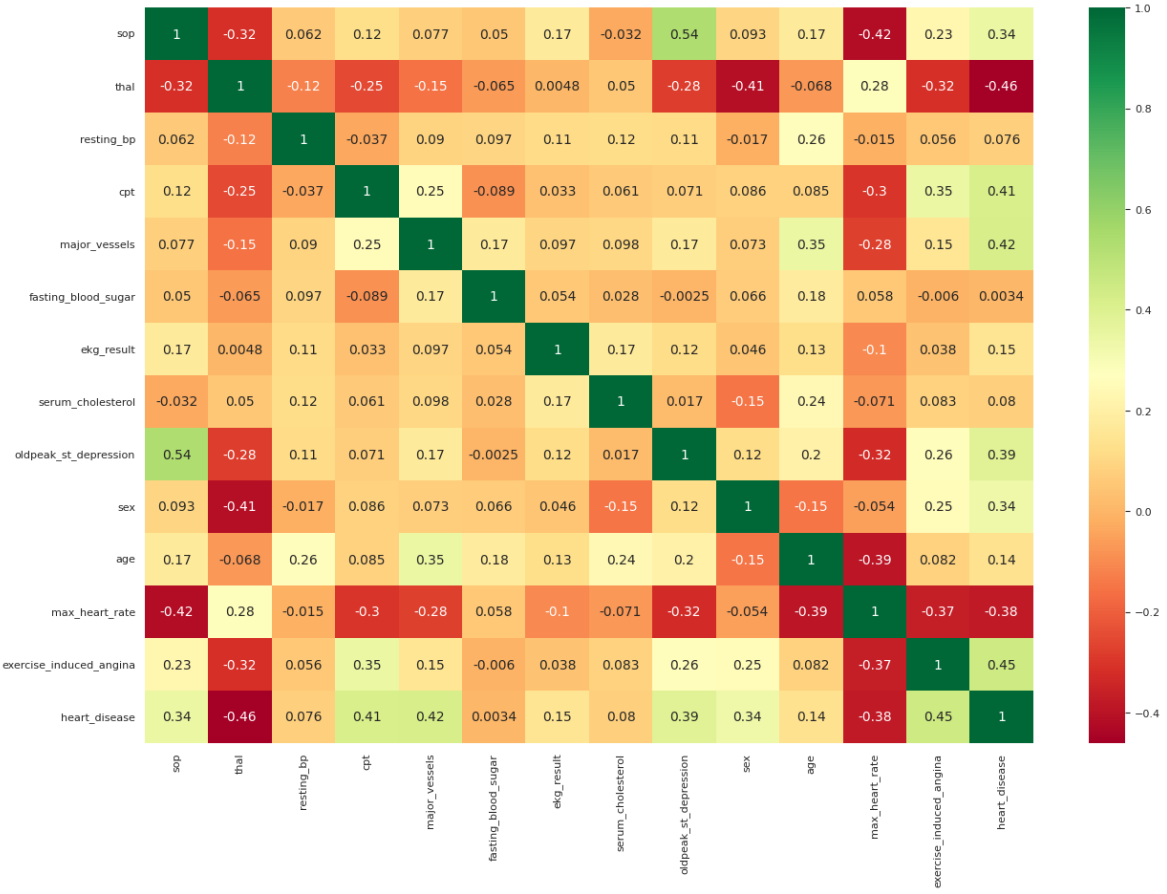
2.CHECKING CORELATION

In [45]:
```
data.corr()
```

Out[45]:

	sop	thal	resting_bp	cpt	major_vessels
sop	1.000000	-0.317019	0.061536	0.121207	0.076832
thal	-0.317019	1.000000	-0.122475	-0.254939	-0.153024
resting_bp	0.061536	-0.122475	1.000000	-0.037038	0.090367
cpt	0.121207	-0.254939	-0.037038	1.000000	0.249061
major_vessels	0.076832	-0.153024	0.090367	0.249061	1.000000
fasting_blood_sugar	0.050199	-0.064897	0.096842	-0.088992	0.169792
ekg_result	0.172191	0.004791	0.113544	0.033379	0.096656
serum_cholesterol	-0.032348	0.050466	0.119426	0.061213	0.098348
oldpeak_st_depression	0.535583	-0.279462	0.110505	0.070715	0.167406
sex	0.093340	-0.412284	-0.016618	0.086057	0.073107
age	0.169918	-0.067663	0.259479	0.085001	0.347355
max_heart_rate	-0.418102	0.278681	-0.014901	-0.301792	-0.275687
exercise_induced_angina	0.225459	-0.317990	0.056117	0.346266	0.153407
heart_disease	0.344224	-0.460933	0.076048	0.412829	0.421519

In [46]:
```
# Plot hitmap for better visualisation
plt.figure(figsize=(15,10))
sns.heatmap(data.corr(),annot=True,cmap='RdYlGn',annot_kws={'size':10})
plt.show()
```



- In dataset no highly correlated feature is available.

3.CHECKING DUPLICATES

In [47]:
```
data.duplicated().sum()
```

Out[47]:
```
0
```

- No duplicate in data

MODEL CREATION

AIM

1. In heart disease case recall metrix is more important so we need more focus to improve recall score
2. Create sweetspot model (Low bias & Low variance)

HERE WE WILL BE EXPERIMENTING WITH FOUR ALGORITHM

1. Logistic regression
2. KNeighborsClassifier
3. RandomForestClassifier
4. XGBClassifier

DEFINE INDEPENDANT AND DEPENDANT VERIABLE

In [48]:
```
X = data.iloc[:,:-1]
y = data.heart_disease
```

In [49]:
```
# Checking the X
X.head()
```

Out[49]:

	sop	thal	resting_bp	cpt	major_vessels	fasting_blood_sugar	ekg_result	serun
0	1	2	-0.117432	2	0	0	2	
1	2	2	-1.352755	3	0	0	0	
2	1	2	-0.323319	4	3	0	2	
3	1	1	1.529666	4	0	0	0	
4	3	1	0.019826	1	0	0	2	

In [50]:
```
# Check y
y.head()
```

Out[50]:
```
0    0
1    0
2    1
3    1
4    0
Name: heart_disease, dtype: int64
```

CHECKING THE BALANCE OF TARGET VERIABLE

In [51]:
```
y.value_counts()
```

Out[51]:
```
heart_disease
0    100
1     80
Name: count, dtype: int64
```

- No need to balance the data

CREATING TRAINING AND TESTING DATA

In [52]:
```
# importing library
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,rando
```

In [53]:
```
# check the shape of X train, y train, X test and y test
X_train.shape
```

Out[53]:
```
(144, 13)
```

In [54]:
```
X_test.shape
```

Out[54]:
```
(36, 13)
```

In [55]:
```
y_train.shape
```

Out[55]: (144,)

In [56]:
```
y_test.shape
```

Out[56]: (36,)

1.LogisticRegression

In [57]:
```
# Step:1 importing library
from sklearn.linear_model import LogisticRegression

# Step:2 Object creation
log_model = LogisticRegression()

# Step:3 Fitting the training data
log_model.fit(X_train,y_train)

# Step:4 Prediction on test data
y_log_predict = log_model.predict(X_test)

# Step:5 Prediction on training data
train_log_predict = log_model.predict(X_train)
```

EVALUATION

TRAINING ACCURACY

In [58]:
```
# importing library
from sklearn.metrics import accuracy_score,recall_score,f1_score,classifi
log_train_accuracy = accuracy_score(train_log_predict,y_train)
print("Training accuracy of Logistic regression model",log_train_accuracy
print("Logistic regression training Classification report: \n",classifica
```

```
Training accuracy of Logistic regression model 84.72222222222221
Logistic regression training Classification report:
               precision    recall  f1-score   support

           0       0.87      0.85      0.86        78
           1       0.82      0.85      0.84        66

    accuracy                           0.85       144
   macro avg       0.85      0.85      0.85       144
weighted avg       0.85      0.85      0.85       144
```

TESTING ACCURACY

In [59]:
```
log_test_accuracy = accuracy_score(y_log_predict,y_test)
print("Testing accuracy ogof Logistic regession model",log_test_accuracy*
print("Logistic regression testing Classification report: \n",classificat
```

```
Testing accuracy ogof Logistic regession model 83.33333333333334
Logistic regression testing Classification report:
              precision    recall  f1-score   support

           0       0.96      0.82      0.88        28
           1       0.58      0.88      0.70         8

    accuracy                           0.83        36
   macro avg       0.77      0.85      0.79        36
weighted avg       0.88      0.83      0.84        36
```

In [60]:
```
# In heaert disease prediction recall is more important
recall_log = recall_score(y_log_predict,y_test)
recall_log*100
```

Out[60]: 87.5

In [61]:
```
# crosstab of logistic regression
pd.crosstab(y_log_predict,y_test)
```

Out[61]:

heart_disease	0	1
row_0		
0	23	5
1	1	7

USE BAGGING ON LOGISTIC REGRESSION MODEL

In [62]:
```
# Step:1 Create logistic regression object
log_reg1 = LogisticRegression()

# Step:2 importing library and creating bagging object
from sklearn.ensemble import BaggingClassifier
bagg = BaggingClassifier(log_reg1,n_estimators=45)
#base_estimator---> algorithm which you want to pass
#n_estimotors-----> number of base learners

# Step:3 Fitting the training data
bagg.fit(X_train,y_train)

# Step:4 Prediction on test data
bagg_predict = bagg.predict(X_test)
```

EVALUATION

In [63]:
```
bagg_recall = recall_score(bagg_predict,y_test)
print("Racall score after bagging",bagg_recall*100)
```

```
Racall score after bagging 87.5
```

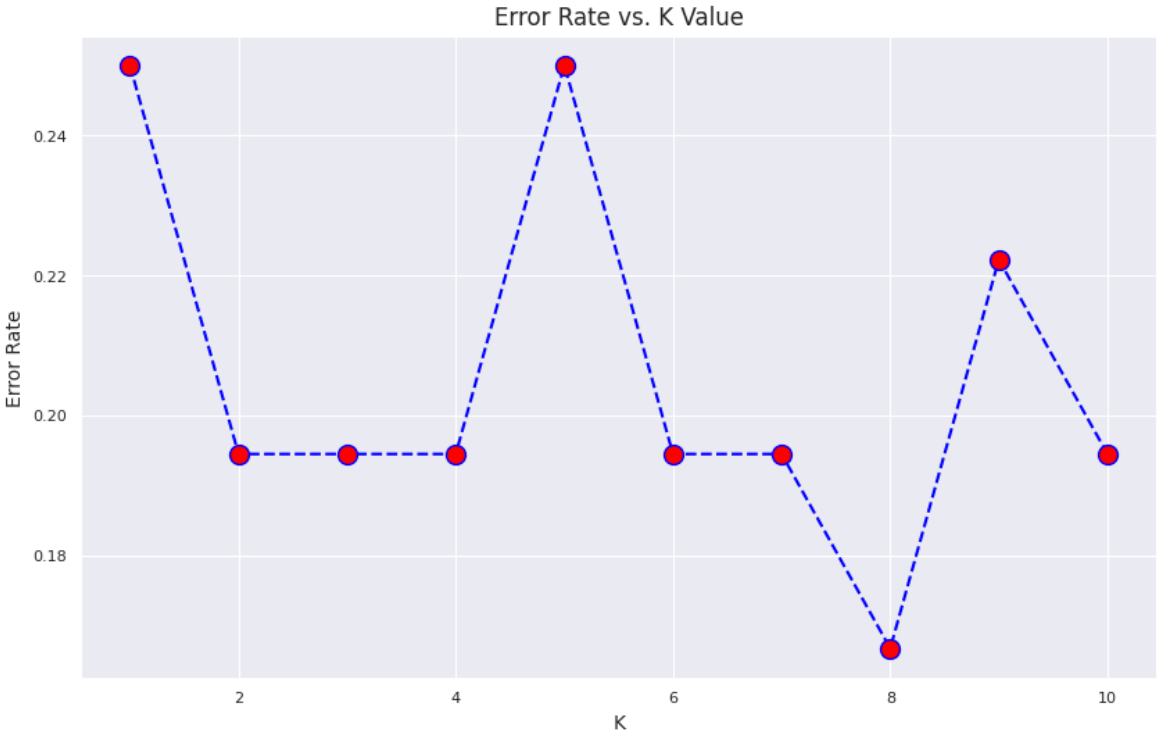
2.KNeighborsClassifier

In [64]:
```
# Step:1 Taking the optimal value of k
from sklearn.neighbors import KNeighborsClassifier
```

```
error_rate = [] # Creating empty list
for i in range(1,11):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    y_knn_predict = knn.predict(X_test)
    error_rate.append(np.mean(y_knn_predict != y_test))
print("Error rate:",error_rate)
```

```
Error rate: [0.25, 0.19444444444444445, 0.19444444444444445, 0.19444444444
444445, 0.25, 0.19444444444444445, 0.19444444444444445, 0.1666666666666666
6, 0.2222222222222222, 0.19444444444444445]
```

In [65]:
```
# Step:2 Plotting the error rate
plt.figure(figsize=(10,6))
plt.plot(range(1,11),error_rate,color='blue', linestyle='dashed',marker='
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
plt.show()
```



In [66]:
```
# Step:3 Get nearst neighbors
knn1 = KNeighborsClassifier(n_neighbors=3)

# Step:4 Fitting the training data
knn1.fit(X_train,y_train)

# Step:5 Prediction on testing data
knn_test_predict = knn1.predict(X_test)

# Step:6 Prediction on training data
knn_train_predict = knn1.predict(X_train)
```

EVALUATION

TRAINING ACCURACY

In [67]:
```
knn_train_accuracy = accuracy_score(knn_train_predict,y_train)
print("Training accuracy of knn model ",knn_train_accuracy)
print("Training classification report:\n",classification_report(knn_train
```

```
Training accuracy of knn model  0.8819444444444444
Training classification report:
               precision    recall  f1-score   support

           0       0.89      0.88      0.89        77
           1       0.87      0.88      0.87        67

    accuracy                           0.88       144
   macro avg       0.88      0.88      0.88       144
weighted avg       0.88      0.88      0.88       144
```

TESTING ACCURACY

In [68]:
```
knn_test_accuracy = accuracy_score(knn_test_predict,y_test)
print("Testing accuracy of knn model",knn_test_accuracy*100)
print("Testing classification report: \n",classification_report(knn_test_
```

```
Testing accuracy of knn model 80.55555555555556
Testing classification report:
               precision    recall  f1-score   support

           0       0.88      0.84      0.86        25
           1       0.67      0.73      0.70        11

    accuracy                           0.81        36
   macro avg       0.77      0.78      0.78        36
weighted avg       0.81      0.81      0.81        36
```

In [69]:
```
# Recall score
recall_knn = recall_score(knn_test_predict,y_test)
recall_knn*100
```

Out[69]: 72.72727272727273

In [70]:
```
# Cross tab
pd.crosstab(knn_test_predict,y_test)
```

Out[70]:

heart_disease	0	1
row_0		
0	21	4
1	3	8

3.RandomForestClassifier

In [71]:
```
# Step:1 importing libary and creating object
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100) # 100 decision tree

# Step:2 fitting training data
```

```
rf.fit(X_train,y_train)

# Step:3 Prediction on testing data
rf_test_predict = rf.predict(X_test)

# Step:4 Prediction on training data
rf_train_predict = rf.predict(X_train)
```

EVALUATION

TRAINING ACCURACY

In [72]:
```
rf_train_accuracy = accuracy_score(rf_train_predict,y_train)
print("Training accuracy of random forest",rf_train_accuracy)
print("Classification report of training: \n",classification_report(rf_tr
```

```
Training accuracy of random forest 1.0
Classification report of training:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        76
           1       1.00      1.00      1.00        68

    accuracy                           1.00       144
   macro avg       1.00      1.00      1.00       144
weighted avg       1.00      1.00      1.00       144
```

- Random forest model very well work on training data

TESTING ACCURACY

In [73]:
```
rf_test_accuracy = accuracy_score(rf_test_predict,y_test)
print("Testing accuracy of random forest",rf_test_accuracy*100)
print("Classification report of testing: \n",classification_report(rf_tes
```

```
Testing accuracy of random forest 80.55555555555556
Classification report of testing:
               precision    recall  f1-score   support

           0       0.88      0.84      0.86        25
           1       0.67      0.73      0.70        11

    accuracy                           0.81        36
   macro avg       0.77      0.78      0.78        36
weighted avg       0.81      0.81      0.81        36
```

- In testing side model is not perfom well so we need to do hyperparameter training

In [74]:
```
# Cross tab
pd.crosstab(rf_test_predict,y_test)
```

Out[74]:

heart_disease	0	1
row_0		
0	21	4
1	3	8

HYPERPARAMETER TUNNING OF RANDOM FOREST

In [75]:
```
# Step:1 import library and imputation of parameter
from sklearn.model_selection import RandomizedSearchCV
#In random forest we are not used grid search CV because of memory reason

n_estimators = [int(x) for x in np.linspace(start=100 ,stop=2000, num=10)
max_features = ['auto', 'sqrt'] #Max no of feature consider to create dec
max_depth    = [int(x) for x in np.linspace(10,100,num=11)] #Max no of le
max_depth.append(None)
min_samples_split = [2,3,5,10] #Min number of data points placed in a nod
min_samples_leaf  = [1,2,3,4,5]  #Min number of data point allowed in lea

# Step:2 Creating dictionary of paramter
random_grid = {'n_estimators': n_estimators, 'max_features': max_features
               'max_depth': max_depth, 'min_samples_split': min_samples_s
               'min_samples_leaf': min_samples_leaf}

# Step:3 Object creation
rf_clf = RandomForestClassifier(random_state=42) #Provide random state be

# Step:4 Create Random search CV with parameter
rf_cv = RandomizedSearchCV(estimator=rf_clf,scoring='f1',param_distributi
                           n_iter=20,cv=2,verbose=2,random_state=42,n_job

# Step:5 Fitting the training data
rf_cv.fit(X_train,y_train)

# Step:6 Get best parameter
rf_best_params = rf_cv.best_params_
print(f"Best parameter: {rf_best_params}")
```

```
Fitting 2 folds for each of 20 candidates, totalling 40 fits
Best parameter: {'n_estimators': 2000, 'min_samples_split': 2, 'min_sample
s_leaf': 5, 'max_features': 'sqrt', 'max_depth': 19}
```

In [76]:
```
# Step:7 Create object and place the best paramter
rf_clf1 = RandomForestClassifier(**rf_best_params)

# Step:8 Fitting the training data
rf_clf1.fit(X_train,y_train)

# Step:9 Prediction on test data
rf_clf1_predict = rf_clf1.predict(X_test)
```

EVALUATION

In [77]:
```
rf_accuracy = accuracy_score(rf_clf1_predict,y_test)
print("Accuracy after hyperparameter tunning",rf_accuracy*100)
```

```
print("Classification report: \n",classification_report(rf_clf1_predict,y
```

```
Accuracy after hyperparameter tunning 80.55555555555556
Classification report:
               precision    recall  f1-score   support

           0       0.92      0.81      0.86        27
           1       0.58      0.78      0.67         9

    accuracy                           0.81        36
   macro avg       0.75      0.80      0.76        36
weighted avg       0.83      0.81      0.81        36
```

In [78]:
```
rf_recall_score = recall_score(rf_clf1_predict,y_test)
print("Recall score:",rf_recall_score*100)
```

```
Recall score: 77.77777777777779
```

4.XGBClassifier

In [79]:
```
# Step:1 import library and object creation
import xgboost
from xgboost import XGBClassifier
xgb = XGBClassifier()

# Step:2 Fitting the training data
xgb.fit(X_train,y_train)

# Step:3 Prediction on training data
xgb_train_predict = xgb.predict(X_train)

# Step:4 Prediction on testing data
xgb_test_predict = xgb.predict(X_test)
```

EVALUATION

TRAINING ACCURACY

In [80]:
```
xgb_train_accuracy = accuracy_score(xgb_train_predict,y_train)
print("Training accuracy of xgb model",xgb_train_accuracy)
print("Classifiaction report on training: \n",classification_report(xgb_t
```

```
Training accuracy of xgb model 1.0
Classifiaction report on training:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        76
           1       1.00      1.00      1.00        68

    accuracy                           1.00       144
   macro avg       1.00      1.00      1.00       144
weighted avg       1.00      1.00      1.00       144
```

- XG boost model is very well work on training data

TESTING ACCURACY

In [81]:
```
xgb_test_accuracy = accuracy_score(xgb_test_predict,y_test)
print("Testing accuracy of xgb model",xgb_test_accuracy*100)
print("Classification report on testing: \n",classification_report(xgb_te
```

```
Testing accuracy of xgb model 77.77777777777779
Classification report on testing:
               precision    recall  f1-score   support

           0       0.88      0.81      0.84        26
           1       0.58      0.70      0.64        10

    accuracy                           0.78        36
   macro avg       0.73      0.75      0.74        36
weighted avg       0.79      0.78      0.78        36
```

In [82]:
```
xgb_recall_score = recall_score(xgb_test_predict,y_test)
print("Recall score:",xgb_recall_score*100)
```

```
Recall score: 70.0
```

- we can see cleary see the model is not perform well on testing data

HYPERPARAMETER TUNNING OF XG BOOST

In [83]:
```
# Step:1 import library
from sklearn.model_selection import RandomizedSearchCV
param_grid = {'gamma': [0,0.1,0.2,0.4,0.8,1.6,3.2,6.4,12.8,25.6,51.2,102.
              'learning_rate': [0.001,0.01,0.1, 0.03, 0.06, 0.1, 0.15, 0.
              'max_depth': [5,6,7,8,9,10,11,12,13,14],
              'n_estimators': [50,65,80,100,115,130,150],
              'reg_alpha': [0,0.01,0.1,0.02,0.2,0.4,0.8,1.6,3.2,6.4,12.8,
              'reg_lambda': [0,0.01,0.1,0.02,0.2,0.4,0.8,1.6,3.2,6.4,12.8

# Step:2 Object creation with parameter
XGB = XGBClassifier(random_state=42)

# Step:3 Create randomized search cv with parameter
rcv = RandomizedSearchCV(estimator=XGB,scoring='f1',param_distributions=p
                         cv=4,verbose=2,random_state=42,n_jobs=-1)

#estimator--number of decision tree
#scoring--->performance matrix to check performance
#param_distribution-->hyperparametes(dictionary we created)
#n_iter--->Number of parameter settings that are sampled. n_iter trades o
##cv------> number of flods
#verbose=Controls the verbosity: the higher, the more messages.
#n_jobs---->Number of jobs to run in parallel,-1 means using all processo

# Step:4 Fitting training data on randomized search cv
rcv.fit(X_train,y_train)

# Step:5 Get best parameters
rcv_best_parameter = rcv.best_params_
print(f"Best parameter: {rcv_best_parameter}")
```

```
Fitting 4 folds for each of 60 candidates, totalling 240 fits
[CV] END max_depth=91, max_features=auto, min_samples_leaf=5, min_samples_
split=3, n_estimators=522; total time=   0.0s
[CV] END max_depth=19, max_features=auto, min_samples_leaf=2, min_samples_
split=5, n_estimators=1366; total time=   0.0s
[CV] END max_depth=19, max_features=auto, min_samples_leaf=2, min_samples_
split=5, n_estimators=1366; total time=   0.0s
[CV] END max_depth=None, max_features=auto, min_samples_leaf=1, min_sample
s_split=5, n_estimators=1366; total time=   0.0s
[CV] END max_depth=82, max_features=sqrt, min_samples_leaf=2, min_samples_
split=2, n_estimators=944; total time=   2.0s
[CV] END max_depth=55, max_features=sqrt, min_samples_leaf=5, min_samples_
split=10, n_estimators=311; total time=   0.6s
[CV] END max_depth=64, max_features=auto, min_samples_leaf=1, min_samples_
split=10, n_estimators=733; total time=   0.0s
[CV] END max_depth=64, max_features=auto, min_samples_leaf=1, min_samples_
split=10, n_estimators=733; total time=   0.0s
[CV] END max_depth=28, max_features=sqrt, min_samples_leaf=5, min_samples_
split=5, n_estimators=944; total time=   2.0s
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_sample
s_split=10, n_estimators=1155; total time=   0.0s
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_sample
s_split=10, n_estimators=1155; total time=   0.0s
[CV] END max_depth=82, max_features=auto, min_samples_leaf=5, min_samples_
split=5, n_estimators=1155; total time=   0.0s
[CV] END max_depth=82, max_features=auto, min_samples_leaf=5, min_samples_
split=5, n_estimators=1155; total time=   0.0s
[CV] END max_depth=100, max_features=auto, min_samples_leaf=3, min_samples
_split=10, n_estimators=1577; total time=   0.0s
[CV] END max_depth=100, max_features=auto, min_samples_leaf=3, min_samples
_split=10, n_estimators=1577; total time=   0.0s
[CV] END max_depth=73, max_features=auto, min_samples_leaf=3, min_samples_
split=5, n_estimators=944; total time=   0.0s
[CV] END max_depth=73, max_features=auto, min_samples_leaf=3, min_samples_
split=5, n_estimators=944; total time=   0.0s
[CV] END max_depth=19, max_features=auto, min_samples_leaf=2, min_samples_
split=10, n_estimators=944; total time=   0.0s
[CV] END max_depth=19, max_features=auto, min_samples_leaf=2, min_samples_
split=10, n_estimators=944; total time=   0.0s
[CV] END max_depth=28, max_features=sqrt, min_samples_leaf=3, min_samples_
split=2, n_estimators=522; total time=   1.1s
[CV] END gamma=12.8, learning_rate=0.06, max_depth=13, n_estimators=150, r
eg_alpha=12.8, reg_lambda=12.8; total time=   0.1s
[CV] END gamma=51.2, learning_rate=0.1, max_depth=12, n_estimators=115, re
g_alpha=3.2, reg_lambda=51.2; total time=   0.1s
[CV] END gamma=12.8, learning_rate=0.1, max_depth=10, n_estimators=80, reg
_alpha=0, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=12.8, learning_rate=0.1, max_depth=10, n_estimators=80, reg
_alpha=0, reg_lambda=6.4; total time=   0.1s
[CV] END gamma=3.2, learning_rate=0.1, max_depth=9, n_estimators=130, reg_
alpha=0.2, reg_lambda=3.2; total time=   0.1s
[CV] END gamma=3.2, learning_rate=0.1, max_depth=9, n_estimators=130, reg_
alpha=0.2, reg_lambda=3.2; total time=   0.1s
[CV] END gamma=102.4, learning_rate=0.7, max_depth=7, n_estimators=50, reg
_alpha=200, reg_lambda=0.2; total time=   0.0s
[CV] END gamma=102.4, learning_rate=0.7, max_depth=7, n_estimators=50, reg
_alpha=200, reg_lambda=0.2; total time=   0.0s
[CV] END gamma=102.4, learning_rate=0.7, max_depth=7, n_estimators=50, reg
_alpha=200, reg_lambda=0.2; total time=   0.0s
[CV] END gamma=102.4, learning_rate=0.7, max_depth=7, n_estimators=50, reg
```

```
_alpha=200, reg_lambda=0.2; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.5, max_depth=11, n_estimators=130, reg
_alpha=51.2, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=1.6, learning_rate=0.5, max_depth=11, n_estimators=130, reg
_alpha=51.2, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=1.6, learning_rate=0.5, max_depth=11, n_estimators=130, reg
_alpha=51.2, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=1.6, learning_rate=0.5, max_depth=11, n_estimators=130, reg
_alpha=51.2, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=0.1, learning_rate=0.001, max_depth=11, n_estimators=115, r
eg_alpha=0.2, reg_lambda=1.6; total time=   0.1s
[CV] END gamma=0.1, learning_rate=0.001, max_depth=11, n_estimators=115, r
eg_alpha=0.2, reg_lambda=1.6; total time=   0.1s
[CV] END gamma=0.1, learning_rate=0.001, max_depth=11, n_estimators=115, r
eg_alpha=0.2, reg_lambda=1.6; total time=   0.1s
[CV] END gamma=0.1, learning_rate=0.001, max_depth=11, n_estimators=115, r
eg_alpha=0.2, reg_lambda=1.6; total time=   0.1s
[CV] END gamma=3.2, learning_rate=0.0003, max_depth=6, n_estimators=130, r
eg_alpha=51.2, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=3.2, learning_rate=0.0003, max_depth=6, n_estimators=130, r
eg_alpha=51.2, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=3.2, learning_rate=0.0003, max_depth=6, n_estimators=130, r
eg_alpha=51.2, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=3.2, learning_rate=0.0003, max_depth=6, n_estimators=130, r
eg_alpha=51.2, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=0.2, learning_rate=0.7, max_depth=7, n_estimators=115, reg_
alpha=51.2, reg_lambda=102.4; total time=   0.0s
[CV] END gamma=0.2, learning_rate=0.7, max_depth=7, n_estimators=115, reg_
alpha=51.2, reg_lambda=102.4; total time=   0.0s
[CV] END gamma=0.2, learning_rate=0.7, max_depth=7, n_estimators=115, reg_
alpha=51.2, reg_lambda=102.4; total time=   0.0s
[CV] END gamma=0.2, learning_rate=0.7, max_depth=7, n_estimators=115, reg_
alpha=51.2, reg_lambda=102.4; total time=   0.0s
[CV] END gamma=0, learning_rate=0.15, max_depth=10, n_estimators=115, reg_
alpha=0.4, reg_lambda=0.4; total time=   0.1s
[CV] END gamma=0, learning_rate=0.15, max_depth=10, n_estimators=115, reg_
alpha=0.4, reg_lambda=0.4; total time=   0.1s
[CV] END gamma=0, learning_rate=0.15, max_depth=10, n_estimators=115, reg_
alpha=0.4, reg_lambda=0.4; total time=   0.1s
[CV] END gamma=0, learning_rate=0.15, max_depth=10, n_estimators=115, reg_
alpha=0.4, reg_lambda=0.4; total time=   0.1s
[CV] END gamma=1.6, learning_rate=0.03, max_depth=11, n_estimators=80, reg
_alpha=3.2, reg_lambda=0.4; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.03, max_depth=11, n_estimators=80, reg
_alpha=3.2, reg_lambda=0.4; total time=   0.1s
[CV] END gamma=1.6, learning_rate=0.03, max_depth=11, n_estimators=80, reg
_alpha=3.2, reg_lambda=0.4; total time=   0.1s
[CV] END gamma=1.6, learning_rate=0.03, max_depth=11, n_estimators=80, reg
_alpha=3.2, reg_lambda=0.4; total time=   0.1s
[CV] END gamma=25.6, learning_rate=0.3, max_depth=5, n_estimators=65, reg_
alpha=12.8, reg_lambda=6.4; total time=   0.1s
[CV] END gamma=25.6, learning_rate=0.3, max_depth=5, n_estimators=65, reg_
alpha=12.8, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=25.6, learning_rate=0.3, max_depth=5, n_estimators=65, reg_
alpha=12.8, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=25.6, learning_rate=0.3, max_depth=5, n_estimators=65, reg_
alpha=12.8, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=200, learning_rate=0.1, max_depth=7, n_estimators=65, reg_a
lpha=1.6, reg_lambda=0.01; total time=   0.0s
[CV] END gamma=200, learning_rate=0.1, max_depth=7, n_estimators=65, reg_a
```

```
lpha=1.6, reg_lambda=0.01; total time=   0.0s
[CV] END gamma=200, learning_rate=0.1, max_depth=7, n_estimators=65, reg_a
lpha=1.6, reg_lambda=0.01; total time=   0.0s
[CV] END gamma=200, learning_rate=0.1, max_depth=7, n_estimators=65, reg_a
lpha=1.6, reg_lambda=0.01; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.0003, max_depth=7, n_estimators=150, r
eg_alpha=51.2, reg_lambda=51.2; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.0003, max_depth=7, n_estimators=150, r
eg_alpha=51.2, reg_lambda=51.2; total time=   0.1s
[CV] END gamma=1.6, learning_rate=0.0003, max_depth=7, n_estimators=150, r
eg_alpha=51.2, reg_lambda=51.2; total time=   0.1s
[CV] END gamma=1.6, learning_rate=0.0003, max_depth=7, n_estimators=150, r
eg_alpha=51.2, reg_lambda=51.2; total time=   0.1s
[CV] END gamma=0, learning_rate=0.1, max_depth=8, n_estimators=130, reg_al
pha=0.02, reg_lambda=6.4; total time=   0.1s
[CV] END max_depth=28, max_features=auto, min_samples_leaf=2, min_samples_
split=5, n_estimators=100; total time=   0.0s
[CV] END max_depth=73, max_features=sqrt, min_samples_leaf=3, min_samples_
split=3, n_estimators=522; total time=   1.1s
[CV] END max_depth=73, max_features=sqrt, min_samples_leaf=5, min_samples_
split=3, n_estimators=311; total time=   0.7s
[CV] END max_depth=19, max_features=sqrt, min_samples_leaf=5, min_samples_
split=2, n_estimators=2000; total time=   4.2s
[CV] END gamma=12.8, learning_rate=0.06, max_depth=13, n_estimators=150, r
eg_alpha=12.8, reg_lambda=12.8; total time=   0.1s
[CV] END gamma=51.2, learning_rate=0.1, max_depth=12, n_estimators=115, re
g_alpha=3.2, reg_lambda=51.2; total time=   0.1s
[CV] END gamma=12.8, learning_rate=0.4, max_depth=11, n_estimators=50, reg
_alpha=3.2, reg_lambda=12.8; total time=   0.1s
[CV] END gamma=12.8, learning_rate=0.4, max_depth=11, n_estimators=50, reg
_alpha=3.2, reg_lambda=12.8; total time=   0.0s
[CV] END gamma=0, learning_rate=0.02, max_depth=5, n_estimators=50, reg_al
pha=0.01, reg_lambda=0.02; total time=   0.1s
[CV] END gamma=0, learning_rate=0.02, max_depth=5, n_estimators=50, reg_al
pha=0.01, reg_lambda=0.02; total time=   0.1s
[CV] END gamma=12.8, learning_rate=0.1, max_depth=13, n_estimators=130, re
g_alpha=0, reg_lambda=200; total time=   0.1s
[CV] END gamma=12.8, learning_rate=0.1, max_depth=13, n_estimators=130, re
g_alpha=0, reg_lambda=200; total time=   0.1s
[CV] END gamma=12.8, learning_rate=0.1, max_depth=13, n_estimators=130, re
g_alpha=0, reg_lambda=200; total time=   0.1s
[CV] END gamma=12.8, learning_rate=0.1, max_depth=13, n_estimators=130, re
g_alpha=0, reg_lambda=200; total time=   0.1s
[CV] END gamma=0.8, learning_rate=0.06, max_depth=6, n_estimators=80, reg_
alpha=0.02, reg_lambda=0.2; total time=   0.1s
[CV] END gamma=0.8, learning_rate=0.06, max_depth=6, n_estimators=80, reg_
alpha=0.02, reg_lambda=0.2; total time=   0.0s
[CV] END gamma=0.8, learning_rate=0.06, max_depth=6, n_estimators=80, reg_
alpha=0.02, reg_lambda=0.2; total time=   0.1s
[CV] END gamma=0.8, learning_rate=0.06, max_depth=6, n_estimators=80, reg_
alpha=0.02, reg_lambda=0.2; total time=   0.0s
[CV] END gamma=0.8, learning_rate=0.0003, max_depth=12, n_estimators=65, r
eg_alpha=12.8, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=0.8, learning_rate=0.0003, max_depth=12, n_estimators=65, r
eg_alpha=12.8, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=0.8, learning_rate=0.0003, max_depth=12, n_estimators=65, r
eg_alpha=12.8, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=0.8, learning_rate=0.0003, max_depth=12, n_estimators=65, r
eg_alpha=12.8, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=0.1, learning_rate=0.03, max_depth=14, n_estimators=50, reg
```

```
_alpha=51.2, reg_lambda=12.8; total time=   0.0s
[CV] END gamma=0.1, learning_rate=0.03, max_depth=14, n_estimators=50, reg
_alpha=51.2, reg_lambda=12.8; total time=   0.1s
[CV] END gamma=0.1, learning_rate=0.03, max_depth=14, n_estimators=50, reg
_alpha=51.2, reg_lambda=12.8; total time=   0.1s
[CV] END gamma=0.1, learning_rate=0.03, max_depth=14, n_estimators=50, reg
_alpha=51.2, reg_lambda=12.8; total time=   0.1s
[CV] END gamma=1.6, learning_rate=0.1, max_depth=5, n_estimators=50, reg_a
lpha=102.4, reg_lambda=12.8; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.1, max_depth=5, n_estimators=50, reg_a
lpha=102.4, reg_lambda=12.8; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.1, max_depth=5, n_estimators=50, reg_a
lpha=102.4, reg_lambda=12.8; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.1, max_depth=5, n_estimators=50, reg_a
lpha=102.4, reg_lambda=12.8; total time=   0.0s
[CV] END gamma=200, learning_rate=0.25, max_depth=8, n_estimators=115, reg
_alpha=6.4, reg_lambda=6.4; total time=   0.1s
[CV] END gamma=200, learning_rate=0.25, max_depth=8, n_estimators=115, reg
_alpha=6.4, reg_lambda=6.4; total time=   0.1s
[CV] END gamma=200, learning_rate=0.25, max_depth=8, n_estimators=115, reg
_alpha=6.4, reg_lambda=6.4; total time=   0.1s
[CV] END gamma=200, learning_rate=0.25, max_depth=8, n_estimators=115, reg
_alpha=6.4, reg_lambda=6.4; total time=   0.1s
[CV] END gamma=12.8, learning_rate=0.25, max_depth=13, n_estimators=130, r
eg_alpha=102.4, reg_lambda=25.6; total time=   0.1s
[CV] END gamma=12.8, learning_rate=0.25, max_depth=13, n_estimators=130, r
eg_alpha=102.4, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=12.8, learning_rate=0.25, max_depth=13, n_estimators=130, r
eg_alpha=102.4, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=12.8, learning_rate=0.25, max_depth=13, n_estimators=130, r
eg_alpha=102.4, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.25, max_depth=8, n_estimators=115, reg
_alpha=1.6, reg_lambda=0.4; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.25, max_depth=8, n_estimators=115, reg
_alpha=1.6, reg_lambda=0.4; total time=   0.1s
[CV] END gamma=1.6, learning_rate=0.25, max_depth=8, n_estimators=115, reg
_alpha=1.6, reg_lambda=0.4; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.25, max_depth=8, n_estimators=115, reg
_alpha=1.6, reg_lambda=0.4; total time=   0.0s
[CV] END gamma=0.4, learning_rate=0.02, max_depth=7, n_estimators=65, reg_
alpha=25.6, reg_lambda=0; total time=   0.0s
[CV] END gamma=0.4, learning_rate=0.02, max_depth=7, n_estimators=65, reg_
alpha=25.6, reg_lambda=0; total time=   0.0s
[CV] END gamma=0.4, learning_rate=0.02, max_depth=7, n_estimators=65, reg_
alpha=25.6, reg_lambda=0; total time=   0.0s
[CV] END gamma=0.4, learning_rate=0.02, max_depth=7, n_estimators=65, reg_
alpha=25.6, reg_lambda=0; total time=   0.0s
[CV] END gamma=6.4, learning_rate=0.1, max_depth=13, n_estimators=130, reg
_alpha=0, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=6.4, learning_rate=0.1, max_depth=13, n_estimators=130, reg
_alpha=0, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=6.4, learning_rate=0.1, max_depth=13, n_estimators=130, reg
_alpha=0, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=6.4, learning_rate=0.1, max_depth=13, n_estimators=130, reg
_alpha=0, reg_lambda=25.6; total time=   0.0s
[CV] END gamma=0.2, learning_rate=0.02, max_depth=6, n_estimators=100, reg
_alpha=6.4, reg_lambda=3.2; total time=   0.0s
[CV] END gamma=0.2, learning_rate=0.02, max_depth=6, n_estimators=100, reg
_alpha=6.4, reg_lambda=3.2; total time=   0.1s
[CV] END gamma=0.2, learning_rate=0.02, max_depth=6, n_estimators=100, reg
```

```
_alpha=6.4, reg_lambda=3.2; total time=   0.1s
[CV] END gamma=0.2, learning_rate=0.02, max_depth=6, n_estimators=100, reg
_alpha=6.4, reg_lambda=3.2; total time=   0.1s
[CV] END gamma=200, learning_rate=0.03, max_depth=14, n_estimators=130, re
g_alpha=12.8, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=200, learning_rate=0.03, max_depth=14, n_estimators=130, re
g_alpha=12.8, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=200, learning_rate=0.03, max_depth=14, n_estimators=130, re
g_alpha=12.8, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=200, learning_rate=0.03, max_depth=14, n_estimators=130, re
g_alpha=12.8, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=6.4, learning_rate=0.1, max_depth=5, n_estimators=150, reg_
alpha=0, reg_lambda=0.2; total time=   0.0s
[CV] END gamma=6.4, learning_rate=0.1, max_depth=5, n_estimators=150, reg_
alpha=0, reg_lambda=0.2; total time=   0.1s
[CV] END gamma=6.4, learning_rate=0.1, max_depth=5, n_estimators=150, reg_
alpha=0, reg_lambda=0.2; total time=   0.1s
[CV] END gamma=6.4, learning_rate=0.1, max_depth=5, n_estimators=150, reg_
alpha=0, reg_lambda=0.2; total time=   0.1s
[CV] END gamma=200, learning_rate=0.7, max_depth=11, n_estimators=65, reg_
alpha=0, reg_lambda=0; total time=   0.0s
[CV] END gamma=200, learning_rate=0.7, max_depth=11, n_estimators=65, reg_
alpha=0, reg_lambda=0; total time=   0.0s
[CV] END gamma=200, learning_rate=0.7, max_depth=11, n_estimators=65, reg_
alpha=0, reg_lambda=0; total time=   0.0s
[CV] END gamma=200, learning_rate=0.7, max_depth=11, n_estimators=65, reg_
alpha=0, reg_lambda=0; total time=   0.0s
[CV] END gamma=12.8, learning_rate=0.5, max_depth=10, n_estimators=65, reg
_alpha=0.8, reg_lambda=0; total time=   0.0s
[CV] END max_depth=91, max_features=auto, min_samples_leaf=5, min_samples_
split=3, n_estimators=522; total time=   0.0s
[CV] END max_depth=None, max_features=auto, min_samples_leaf=1, min_sample
s_split=5, n_estimators=1366; total time=   0.0s
[CV] END max_depth=82, max_features=sqrt, min_samples_leaf=2, min_samples_
split=2, n_estimators=944; total time=   2.0s
[CV] END max_depth=55, max_features=sqrt, min_samples_leaf=5, min_samples_
split=10, n_estimators=311; total time=   0.7s
[CV] END max_depth=28, max_features=sqrt, min_samples_leaf=5, min_samples_
split=5, n_estimators=944; total time=   2.0s
[CV] END max_depth=28, max_features=sqrt, min_samples_leaf=3, min_samples_
split=2, n_estimators=522; total time=   1.1s
[CV] END gamma=12.8, learning_rate=0.06, max_depth=13, n_estimators=150, r
eg_alpha=12.8, reg_lambda=12.8; total time=   0.0s
[CV] END gamma=51.2, learning_rate=0.1, max_depth=12, n_estimators=115, re
g_alpha=3.2, reg_lambda=51.2; total time=   0.0s
[CV] END gamma=51.2, learning_rate=0.1, max_depth=12, n_estimators=115, re
g_alpha=3.2, reg_lambda=51.2; total time=   0.0s
[CV] END gamma=12.8, learning_rate=0.4, max_depth=11, n_estimators=50, reg
_alpha=3.2, reg_lambda=12.8; total time=   0.0s
[CV] END gamma=12.8, learning_rate=0.4, max_depth=11, n_estimators=50, reg
_alpha=3.2, reg_lambda=12.8; total time=   0.0s
[CV] END gamma=3.2, learning_rate=0.1, max_depth=9, n_estimators=130, reg_
alpha=0.2, reg_lambda=3.2; total time=   0.1s
[CV] END gamma=3.2, learning_rate=0.1, max_depth=9, n_estimators=130, reg_
alpha=0.2, reg_lambda=3.2; total time=   0.0s
[CV] END gamma=0.2, learning_rate=0.04, max_depth=9, n_estimators=150, reg
_alpha=0.1, reg_lambda=0; total time=   0.1s
[CV] END gamma=0.2, learning_rate=0.04, max_depth=9, n_estimators=150, reg
_alpha=0.1, reg_lambda=0; total time=   0.1s
[CV] END gamma=0.2, learning_rate=0.04, max_depth=9, n_estimators=150, reg
```

```
_alpha=0.1, reg_lambda=0; total time=   0.1s
[CV] END gamma=0.2, learning_rate=0.04, max_depth=9, n_estimators=150, reg
_alpha=0.1, reg_lambda=0; total time=   0.1s
[CV] END gamma=6.4, learning_rate=0.25, max_depth=6, n_estimators=150, reg
_alpha=0.8, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=6.4, learning_rate=0.25, max_depth=6, n_estimators=150, reg
_alpha=0.8, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=6.4, learning_rate=0.25, max_depth=6, n_estimators=150, reg
_alpha=0.8, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=6.4, learning_rate=0.25, max_depth=6, n_estimators=150, reg
_alpha=0.8, reg_lambda=6.4; total time=   0.0s
[CV] END gamma=12.8, learning_rate=0.25, max_depth=7, n_estimators=130, re
g_alpha=0.4, reg_lambda=0.4; total time=   0.0s
[CV] END gamma=12.8, learning_rate=0.25, max_depth=7, n_estimators=130, re
g_alpha=0.4, reg_lambda=0.4; total time=   0.0s
[CV] END gamma=12.8, learning_rate=0.25, max_depth=7, n_estimators=130, re
g_alpha=0.4, reg_lambda=0.4; total time=   0.0s
[CV] END gamma=12.8, learning_rate=0.25, max_depth=7, n_estimators=130, re
g_alpha=0.4, reg_lambda=0.4; total time=   0.0s
[CV] END gamma=102.4, learning_rate=0.7, max_depth=13, n_estimators=100, r
eg_alpha=0.01, reg_lambda=3.2; total time=   0.0s
[CV] END gamma=102.4, learning_rate=0.7, max_depth=13, n_estimators=100, r
eg_alpha=0.01, reg_lambda=3.2; total time=   0.0s
[CV] END gamma=102.4, learning_rate=0.7, max_depth=13, n_estimators=100, r
eg_alpha=0.01, reg_lambda=3.2; total time=   0.0s
[CV] END gamma=102.4, learning_rate=0.7, max_depth=13, n_estimators=100, r
eg_alpha=0.01, reg_lambda=3.2; total time=   0.0s
[CV] END gamma=0.2, learning_rate=0.6, max_depth=14, n_estimators=130, reg
_alpha=12.8, reg_lambda=0.01; total time=   0.0s
[CV] END gamma=0.2, learning_rate=0.6, max_depth=14, n_estimators=130, reg
_alpha=12.8, reg_lambda=0.01; total time=   0.0s
[CV] END gamma=0.2, learning_rate=0.6, max_depth=14, n_estimators=130, reg
_alpha=12.8, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=0.2, learning_rate=0.6, max_depth=14, n_estimators=130, reg
_alpha=12.8, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=3.2, learning_rate=0.4, max_depth=14, n_estimators=50, reg_
alpha=6.4, reg_lambda=102.4; total time=   0.0s
[CV] END gamma=3.2, learning_rate=0.4, max_depth=14, n_estimators=50, reg_
alpha=6.4, reg_lambda=102.4; total time=   0.1s
[CV] END gamma=3.2, learning_rate=0.4, max_depth=14, n_estimators=50, reg_
alpha=6.4, reg_lambda=102.4; total time=   0.1s
[CV] END gamma=3.2, learning_rate=0.4, max_depth=14, n_estimators=50, reg_
alpha=6.4, reg_lambda=102.4; total time=   0.1s
[CV] END gamma=200, learning_rate=0.0003, max_depth=11, n_estimators=150,
reg_alpha=200, reg_lambda=200; total time=   0.1s
[CV] END gamma=200, learning_rate=0.0003, max_depth=11, n_estimators=150,
reg_alpha=200, reg_lambda=200; total time=   0.1s
[CV] END gamma=200, learning_rate=0.0003, max_depth=11, n_estimators=150,
reg_alpha=200, reg_lambda=200; total time=   0.1s
[CV] END gamma=200, learning_rate=0.0003, max_depth=11, n_estimators=150,
reg_alpha=200, reg_lambda=200; total time=   0.1s
[CV] END gamma=0.4, learning_rate=0.3, max_depth=13, n_estimators=65, reg_
alpha=1.6, reg_lambda=0.8; total time=   0.1s
[CV] END gamma=0.4, learning_rate=0.3, max_depth=13, n_estimators=65, reg_
alpha=1.6, reg_lambda=0.8; total time=   0.1s
[CV] END gamma=0.4, learning_rate=0.3, max_depth=13, n_estimators=65, reg_
alpha=1.6, reg_lambda=0.8; total time=   0.1s
[CV] END gamma=0.4, learning_rate=0.3, max_depth=13, n_estimators=65, reg_
alpha=1.6, reg_lambda=0.8; total time=   0.1s
[CV] END gamma=1.6, learning_rate=0.4, max_depth=8, n_estimators=80, reg_a
```

```
lpha=0.4, reg_lambda=51.2; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.4, max_depth=8, n_estimators=80, reg_a
lpha=0.4, reg_lambda=51.2; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.4, max_depth=8, n_estimators=80, reg_a
lpha=0.4, reg_lambda=51.2; total time=   0.0s
[CV] END gamma=1.6, learning_rate=0.4, max_depth=8, n_estimators=80, reg_a
lpha=0.4, reg_lambda=51.2; total time=   0.0s
[CV] END gamma=0.2, learning_rate=0.02, max_depth=10, n_estimators=130, re
g_alpha=0.2, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=0.2, learning_rate=0.02, max_depth=10, n_estimators=130, re
g_alpha=0.2, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=0.2, learning_rate=0.02, max_depth=10, n_estimators=130, re
g_alpha=0.2, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=0.2, learning_rate=0.02, max_depth=10, n_estimators=130, re
g_alpha=0.2, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=0.4, learning_rate=0.003, max_depth=10, n_estimators=65, re
g_alpha=25.6, reg_lambda=51.2; total time=   0.0s
[CV] END gamma=0.4, learning_rate=0.003, max_depth=10, n_estimators=65, re
g_alpha=25.6, reg_lambda=51.2; total time=   0.0s
[CV] END gamma=0.4, learning_rate=0.003, max_depth=10, n_estimators=65, re
g_alpha=25.6, reg_lambda=51.2; total time=   0.0s
[CV] END gamma=0.4, learning_rate=0.003, max_depth=10, n_estimators=65, re
g_alpha=25.6, reg_lambda=51.2; total time=   0.1s
[CV] END gamma=51.2, learning_rate=0.001, max_depth=12, n_estimators=65, r
eg_alpha=3.2, reg_lambda=0.4; total time=   0.1s
[CV] END gamma=51.2, learning_rate=0.001, max_depth=12, n_estimators=65, r
eg_alpha=3.2, reg_lambda=0.4; total time=   0.0s
[CV] END gamma=51.2, learning_rate=0.001, max_depth=12, n_estimators=65, r
eg_alpha=3.2, reg_lambda=0.4; total time=   0.0s
[CV] END gamma=51.2, learning_rate=0.001, max_depth=12, n_estimators=65, r
eg_alpha=3.2, reg_lambda=0.4; total time=   0.0s
[CV] END gamma=25.6, learning_rate=0.6, max_depth=9, n_estimators=100, reg
_alpha=0.01, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=25.6, learning_rate=0.6, max_depth=9, n_estimators=100, reg
_alpha=0.01, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=25.6, learning_rate=0.6, max_depth=9, n_estimators=100, reg
_alpha=0.01, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=25.6, learning_rate=0.6, max_depth=9, n_estimators=100, reg
_alpha=0.01, reg_lambda=0.01; total time=   0.1s
[CV] END gamma=25.6, learning_rate=0.3, max_depth=7, n_estimators=80, reg_
alpha=0.1, reg_lambda=200; total time=   0.1s
[CV] END gamma=25.6, learning_rate=0.3, max_depth=7, n_estimators=80, reg_
alpha=0.1, reg_lambda=200; total time=   0.0s
Best parameter: {'reg_lambda': 102.4, 'reg_alpha': 0.8, 'n_estimators': 80
, 'max_depth': 10, 'learning_rate': 0.7, 'gamma': 0.1}
```

In [84]:
```
# Step:6 Place the best parameter
XGB2 = XGBClassifier(reg_lambda=12.8,reg_alpha=0.4,n_estimators=115,max_d

# Step:7 Fitting the training data
XGB2.fit(X_train,y_train)

# Step:8 Prediction on testing data
XGB_prediction = XGB2.predict(X_test)
```

EVALUATION

In [85]:
```
XGB_accuracy = accuracy_score(XGB_prediction,y_test)
print("Accuracy score after hyperparameter tunning",XGB_accuracy*100)
```

```
print("Classification report: \n",classification_report(XGB_prediction,y_
```

```
Accuracy score after hyperparameter tunning 75.0
Classification report:
               precision    recall  f1-score   support

           0       0.79      0.83      0.81        23
           1       0.67      0.62      0.64        13

    accuracy                           0.75        36
   macro avg       0.73      0.72      0.72        36
weighted avg       0.75      0.75      0.75        36
```

In [86]:
```
XGB_recall = recall_score(XGB_prediction,y_test)
print("Recall Score",XGB_recall*100)
```

```
Recall Score 61.53846153846154
```

CONCLUSION:

1. Logistic regression model is performed well on training data with 84.72% accuracy and testing data with 83.33% accuracy as well as recall score is 87.50% after apply bagging recall score 88.88%.
2. KNN model is performed very well on training data with 88.19% accuracy but in testing data model accuracy and recall score is lagging
3. Random forest model performed very well on training data with 100% accuracy but in testing data model is not perfomed well after applying hyperparameter tunning recall score still lagging
4. XG boost model is also performed well on training data with 100% accuracy but in testing data accuracy is extrmely lagging after apply hyperparameter tunning the recall score is 87.50%
5. From above all model we are select logistic regression model beacuse it is performed very well on training as well as testing data and recall score is also good.

MODEL SAVING

In [87]:
```
# Model saving using pickle
import pickle

file = open("modelPredict.pkl","wb")
# Dump information to file
pickle.dump(log_model,file)
```

In [ ]:
```

```

In [88]:
```
pickle.dump(log_model,file)
```

In [ ]:
```

```