In [4]:
```python
import numpy as np
import pandas as pd
from collections import Counter
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
```

In [5]:
```python
df=pd.read_csv("heart.csv")
```

In [6]:
```python
df
```

Out[6]:

ge	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	C
40	M	ATA	140	289	0	Normal	172	N	
49	F	NAP	160	180	0	Normal	156	N	
37	M	ATA	130	283	0	ST	98	N	
48	F	ASY	138	214	0	Normal	108	Y	
54	M	NAP	150	195	0	Normal	122	N	
...	...	...	...	...	...	...	...	...	
45	M	TA	110	264	0	Normal	132	N	
68	M	ASY	144	193	1	Normal	141	N	
57	M	ASY	130	131	0	Normal	115	Y	
57	F	ATA	130	236	0	LVH	174	N	
38	M	NAP	138	175	0	Normal	173	N	

s × 12 columns

In [7]:
```python
df.shape
```

Out[7]:
```
(918, 12)
```

In [8]:
```python
df.columns
```

Out[8]:
```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

In [11]:
```
df.describe()
```

Out[11]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

In [12]:

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

Out[12]:

```
1    508
0    410
Name: HeartDisease, dtype: int64
```

```
In [13]: pd.crosstab(df['Age'],df['FastingBS'])
```

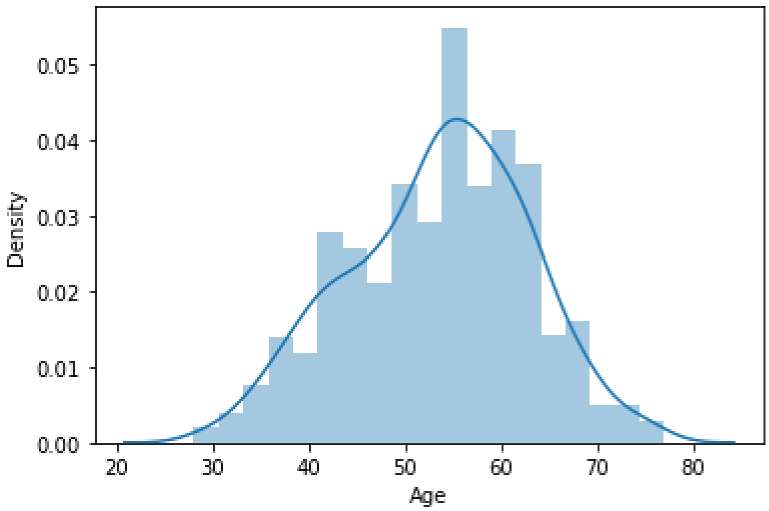
Out[13]:

FastingBS	0	1
Age		
28	1	0
29	3	0
30	1	0
31	2	0
32	4	1
33	2	0
34	6	1
35	10	1
36	5	1
37	11	0
38	13	3
39	14	1
40	10	3
41	22	2
42	16	2
43	21	3
44	19	0
45	17	1
46	21	3
47	15	4
48	23	8
49	21	0
50	23	2
51	23	12
52	27	9
53	24	9
54	43	8
55	33	8
56	24	14
57	25	13
58	30	12
59	26	9
60	21	11

FastingBS	0	1
Age		
61	20	11
62	24	11
63	21	9
64	16	6
65	14	7
66	10	3
67	11	4
68	4	6
69	6	7
70	6	1
71	4	1
72	3	1
73	1	0
74	3	4
75	1	2
76	2	0
77	2	0

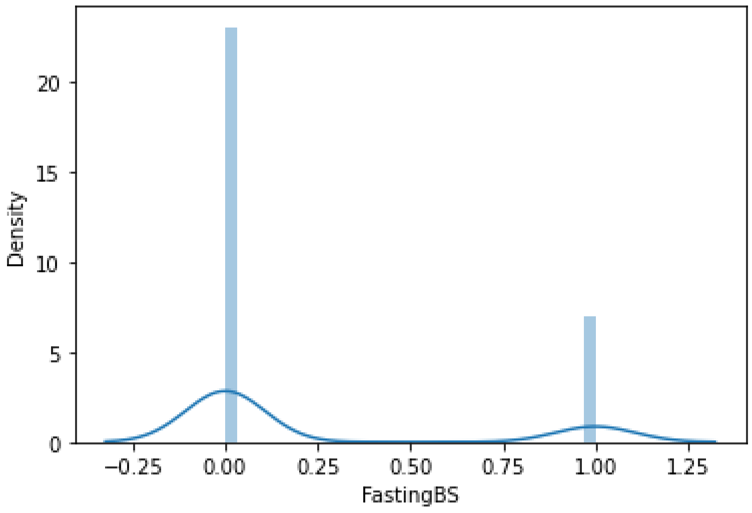
```
In [14]: sns.distplot(df['Age'])
```

```
Out[14]: <AxesSubplot:xlabel='Age', ylabel='Density'>
```



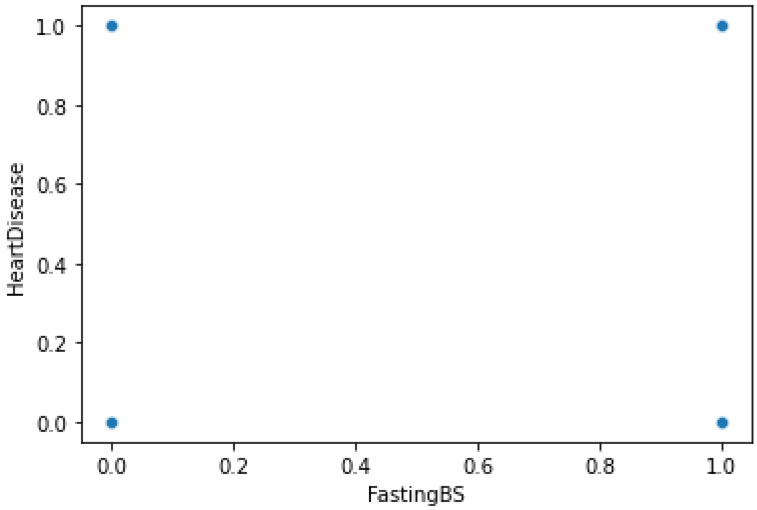
```
In [15]: sns.distplot(df['FastingBS'])
```

```
Out[15]: <AxesSubplot:xlabel='FastingBS', ylabel='Density'>
```



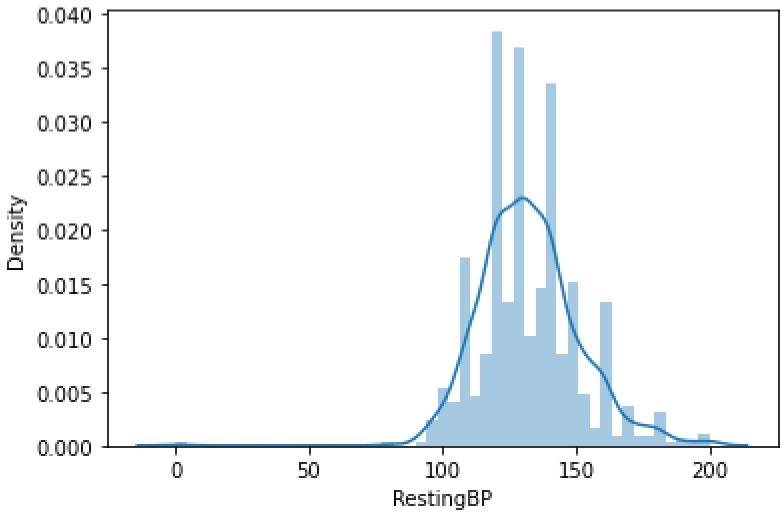
```
In [16]: sns.scatterplot(x='FastingBS',y='HeartDisease',data=df)
```

```
Out[16]: <AxesSubplot:xlabel='FastingBS', ylabel='HeartDisease'>
```



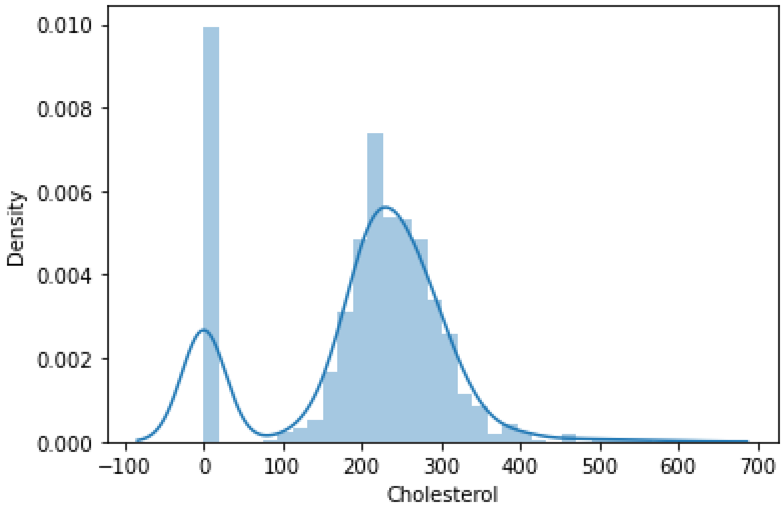
```
In [37]: sns.distplot(df['RestingBP'])
```

```
Out[37]: <AxesSubplot:xlabel='RestingBP', ylabel='Density'>
```



```
In [38]: sns.distplot(df['Cholesterol'])
```

```
Out[38]: <AxesSubplot:xlabel='Cholesterol', ylabel='Density'>
```



In [18]:
```
df.head(20)
```

Out[18]:

Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpe
M	ATA	140	289	0	Normal	172	N	
F	NAP	160	180	0	Normal	156	N	
M	ATA	130	283	0	ST	98	N	
F	ASY	138	214	0	Normal	108	Y	
M	NAP	150	195	0	Normal	122	N	
M	NAP	120	339	0	Normal	170	N	
F	ATA	130	237	0	Normal	170	N	
M	ATA	110	208	0	Normal	142	N	
M	ASY	140	207	0	Normal	130	Y	
F	ATA	120	284	0	Normal	120	N	
F	NAP	130	211	0	Normal	142	N	
M	ATA	136	164	0	ST	99	Y	
M	ATA	120	204	0	Normal	145	N	
M	ASY	140	234	0	Normal	140	Y	
F	NAP	115	211	0	ST	137	N	
F	ATA	120	273	0	Normal	150	N	
M	ASY	110	196	0	Normal	166	N	
F	ATA	120	201	0	Normal	165	N	
M	ASY	100	248	0	Normal	125	N	
M	ATA	120	267	0	Normal	160	N	

In [19]:
```
df['ST_Slope'].value_counts()
```

Out[19]:
```
Flat    460
Up      395
Down     63
Name: ST_Slope, dtype: int64
```

In [60]:
```
df['ST_Slope'].replace(['Flat','Up','Down'],[0,1,-1],inplace=True)
```

In [61]:
```
df["FastingBS"].value_counts()
```

Out[61]:
```
0    704
1    214
Name: FastingBS, dtype: int64
```

```
In [62]: df['Cholesterol'].value_counts()
```

```
Out[62]: 0      172
         254     11
         223     10
         220     10
         230      9
               ...
         392      1
         316      1
         153      1
         466      1
         131      1
         Name: Cholesterol, Length: 222, dtype: int64
```

```
In [67]: df['RestingECG'].value_counts()
```

```
Out[67]: Normal    552
         LVH       188
         ST        178
         Name: RestingECG, dtype: int64
```

```
In [71]: df['RestingECG'].replace(['Normal','LVH','ST'],[0,1,2],inplace=True)
```

```
In [73]: df["ChestPainType"].value_counts()
```

```
Out[73]: ASY    496
         NAP    203
         ATA    173
         TA      46
         Name: ChestPainType, dtype: int64
```

```
In [74]: df['ChestPainType'].replace(['ASY','NAP','ATA','TA'],[0,1,2,3],inplace=True)
```

```
In [77]: df['Sex'].value_counts()
```

```
Out[77]: M    725
         F    193
         Name: Sex, dtype: int64
```

```
In [78]: df['Sex'].replace(['M','F'],[1,0],inplace=True)
```

In [79]:

```
df.head(30)
```

Out[79]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseA
0	40	1	2	140	289	0	0	172	
1	49	0	1	160	180	0	0	156	
2	37	1	2	130	283	0	2	98	
3	48	0	0	138	214	0	0	108	
4	54	1	1	150	195	0	0	122	
5	39	1	1	120	339	0	0	170	
6	45	0	2	130	237	0	0	170	
7	54	1	2	110	208	0	0	142	
8	37	1	0	140	207	0	0	130	
9	48	0	2	120	284	0	0	120	
10	37	0	1	130	211	0	0	142	
11	58	1	2	136	164	0	2	99	
12	39	1	2	120	204	0	0	145	
13	49	1	0	140	234	0	0	140	
14	42	0	1	115	211	0	2	137	
15	54	0	2	120	273	0	0	150	
16	38	1	0	110	196	0	0	166	
17	43	0	2	120	201	0	0	165	
18	60	1	0	100	248	0	0	125	
19	36	1	2	120	267	0	0	160	
20	43	0	3	100	223	0	0	142	
21	44	1	2	120	184	0	0	142	
22	49	0	2	124	201	0	0	164	
23	44	1	2	150	288	0	0	150	
24	40	1	1	130	215	0	0	138	
25	36	1	1	130	209	0	0	178	
26	53	1	0	124	260	0	2	112	
27	52	1	2	120	284	0	0	118	
28	53	0	2	113	468	0	0	127	
29	51	1	2	125	188	0	0	145	

Data Cleaning

In [80]:
```
df.describe()
```

Out[80]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	9
mean	53.510893	0.789760	0.748366	132.396514	198.799564	0.233115	0.592593	1
std	9.432617	0.407701	0.931031	18.514154	109.384145	0.423046	0.793670	
min	28.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	47.000000	1.000000	0.000000	120.000000	173.250000	0.000000	0.000000	1
50%	54.000000	1.000000	0.000000	130.000000	223.000000	0.000000	0.000000	1
75%	60.000000	1.000000	1.000000	140.000000	267.000000	0.000000	1.000000	1
max	77.000000	1.000000	3.000000	200.000000	603.000000	1.000000	2.000000	2

In [81]:
```
df_copy=df.copy(deep=True)
```

In [82]:
```
['RestingBP','Cholesterol','Oldpeak']]=df_copy[['RestingBP','Cholesterol','Oldpeak
```

In [83]:
```
df_copy
```

Out[83]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Exercise/
0	40	1	2	140.0	289.0	0	0	172	
1	49	0	1	160.0	180.0	0	0	156	
2	37	1	2	130.0	283.0	0	2	98	
3	48	0	0	138.0	214.0	0	0	108	
4	54	1	1	150.0	195.0	0	0	122	
...	...	...	...	...	...	...	...	...	
913	45	1	3	110.0	264.0	0	0	132	
914	68	1	0	144.0	193.0	1	0	141	
915	57	1	0	130.0	131.0	0	0	115	
916	57	0	2	130.0	236.0	0	1	174	
917	38	1	1	138.0	175.0	0	0	173	

918 rows × 12 columns

In [84]:
```
df_copy.isnull().sum()
```

Out[84]:
```
Age                 0
Sex                 0
ChestPainType       0
RestingBP           1
Cholesterol       172
FastingBS           0
RestingECG          0
MaxHR               0
ExerciseAngina      0
Oldpeak           368
ST_Slope            0
HeartDisease        0
dtype: int64
```

In [85]:
```
df_copy.describe()
```

Out[85]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	
count	918.000000	918.000000	918.000000	917.000000	746.000000	918.000000	918.000000	9
mean	53.510893	0.789760	0.748366	132.540894	244.635389	0.233115	0.592593	1
std	9.432617	0.407701	0.931031	17.999749	59.153524	0.423046	0.793670	
min	28.000000	0.000000	0.000000	80.000000	85.000000	0.000000	0.000000	
25%	47.000000	1.000000	0.000000	120.000000	207.250000	0.000000	0.000000	1
50%	54.000000	1.000000	0.000000	130.000000	237.000000	0.000000	0.000000	1
75%	60.000000	1.000000	1.000000	140.000000	275.000000	0.000000	1.000000	1
max	77.000000	1.000000	3.000000	200.000000	603.000000	1.000000	2.000000	2

In [86]:
```
#becaues distrubution is very near to Normal Distribution so we can fill the null
df_copy['RestingBP'].fillna(df_copy['RestingBP'].mean(),inplace=True)
```

In [87]:
```
df_copy['Cholesterol'].fillna(df_copy['Cholesterol'].mean(),inplace=True)
```

In [88]:
```
df_copy['Oldpeak'].fillna(df_copy['Oldpeak'].mean(),inplace=True)
```

In [89]:
```
df_copy.isnull().sum()
```

Out[89]:
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

In [90]:
```
df_copy.describe()
```

Out[90]:

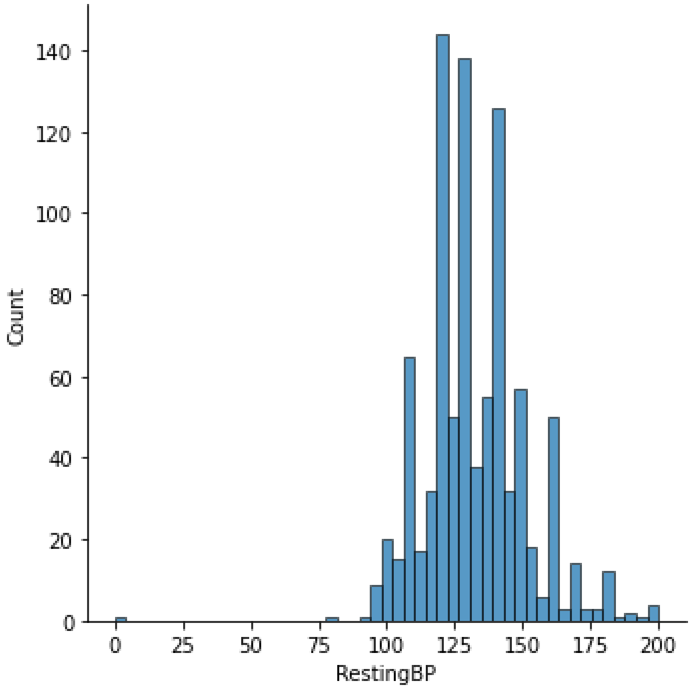
	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	9
mean	53.510893	0.789760	0.748366	132.540894	244.635389	0.233115	0.592593	1
std	9.432617	0.407701	0.931031	17.989932	53.318029	0.423046	0.793670	
min	28.000000	0.000000	0.000000	80.000000	85.000000	0.000000	0.000000	
25%	47.000000	1.000000	0.000000	120.000000	214.000000	0.000000	0.000000	1
50%	54.000000	1.000000	0.000000	130.000000	244.635389	0.000000	0.000000	1
75%	60.000000	1.000000	1.000000	140.000000	267.000000	0.000000	1.000000	1
max	77.000000	1.000000	3.000000	200.000000	603.000000	1.000000	2.000000	2

In [91]:

```
sns.displot(df['RestingBP'])
```

Out[91]:

```
<seaborn.axisgrid.FacetGrid at 0x269f22b9820>
```



In [92]:

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

Out[92]:

```
1    508
0    410
Name: HeartDisease, dtype: int64
```

```
# The distribution is nearly same hence we go directly go through the train
split
```

Train and test split

Train and test split

In [93]:

```
x=df_copy.drop('HeartDisease',axis=1)
```

In [94]:

```
x
```

Out[94]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseA
0	40	1	2	140.0	289.0	0	0	172	
1	49	0	1	160.0	180.0	0	0	156	
2	37	1	2	130.0	283.0	0	2	98	
3	48	0	0	138.0	214.0	0	0	108	
4	54	1	1	150.0	195.0	0	0	122	
...	...	...	...	...	...	...	...	...	
913	45	1	3	110.0	264.0	0	0	132	
914	68	1	0	144.0	193.0	1	0	141	
915	57	1	0	130.0	131.0	0	0	115	
916	57	0	2	130.0	236.0	0	1	174	
917	38	1	1	138.0	175.0	0	0	173	

918 rows × 11 columns

In [95]:

```
y=df_copy['HeartDisease']
```

In [96]:

```
y
```

Out[96]:

```
0       0
1       1
2       0
3       1
4       0
       ..
913     1
914     1
915     1
916     1
917     0
Name: HeartDisease, Length: 918, dtype: int64
```

In [97]:

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
In [98]: X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 642 entries, 112 to 633
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             642 non-null    int64  
 1   Sex             642 non-null    int64  
 2   ChestPainType   642 non-null    int64  
 3   RestingBP       642 non-null    float64
 4   Cholesterol     642 non-null    float64
 5   FastingBS       642 non-null    int64  
 6   RestingECG      642 non-null    int64  
 7   MaxHR           642 non-null    int64  
 8   ExerciseAngina  642 non-null    int64  
 9   Oldpeak         642 non-null    float64
 10  ST_Slope        642 non-null    int64  
dtypes: float64(3), int64(8)
memory usage: 60.2 KB
```

Logistic Regression Algorithms

```
In [99]: from sklearn.linear_model import LogisticRegression
```

```
In [100]: logmodel=LogisticRegression()
```

```
In [101]: logmodel.fit(X_train,y_train)
```

```
Out[101]: LogisticRegression()
```

```
In [102]: predictions=logmodel.predict(X_test)
```

```
In [103]: from sklearn.metrics import classification_report
```

```
In [105]: print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.78      0.86      0.82       114
           1       0.89      0.83      0.86       162

    accuracy                           0.84       276
   macro avg       0.84      0.85      0.84       276
weighted avg       0.85      0.84      0.85       276
```

```
In [110]: from sklearn.metrics import confusion_matrix
In [111]: print(confusion_matrix(y_test,predictions))
```

```
[[ 98  16]
 [ 27 135]]
```

```
In [ ]:
```