Project_04_Heart Failure Prediction

https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction (https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction)

Problem Statement :

With a plethora of medical data available and the rise of Data Science, a host of startups are taking up the challenge of attempting to create indicators for the forseen diseases that might be contracted! Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Heart failure is a common event caused by CVDs. People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help. In this way, we try to solve automate another problem that occurs in the nature with a view to counter it and focus on to the next problem with the help of AI techniques!

Aim :

- To classify / predict whether a patient is prone to heart failure depending on multiple attributes.
- It is a **binary classification** with multiple numerical and categorical features.

Dataset Attributes

- **Age** : age of the patient [years]
- **Sex** : sex of the patient [M: Male, F: Female]
- **ChestPainType** : chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
- **RestingBP** : resting blood pressure [mm Hg]
- **Cholesterol** : serum cholesterol [mm/dl]
- **FastingBS** : fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
- **RestingECG** : resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
- **MaxHR** : maximum heart rate achieved [Numeric value between 60 and 202]
- **ExerciseAngina** : exercise-induced angina [Y: Yes, N: No]
- **Oldpeak** : oldpeak = ST [Numeric value measured in depression]
- **ST_Slope** : the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
- **HeartDisease** : output class [1: heart disease, 0: Normal]

Notebook Contents :

- Dataset Information
- Exploratory Data Analysis (EDA)
- Summary of EDA
- Modeling
- Conclusion

What you will learn :

- Data Visualization
- Data Scaling
- Statistical Tests for Feature Engineering
- Modeling and visualization of results for algorithms

Dataset Information

Import the Necessary Libraries :

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
pd.options.display.float_format = '{:.2f}'.format
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_csv('heart.csv')
data.head()
```

Out[2]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Exercise
0	40	M	ATA	140	289	0	Normal	172	
1	49	F	NAP	160	180	0	Normal	156	
2	37	M	ATA	130	283	0	ST	98	
3	48	F	ASY	138	214	0	Normal	108	
4	54	M	NAP	150	195	0	Normal	122	

In [3]:
```
1 data.shape
```

Out[3]:
```
(918, 12)
```

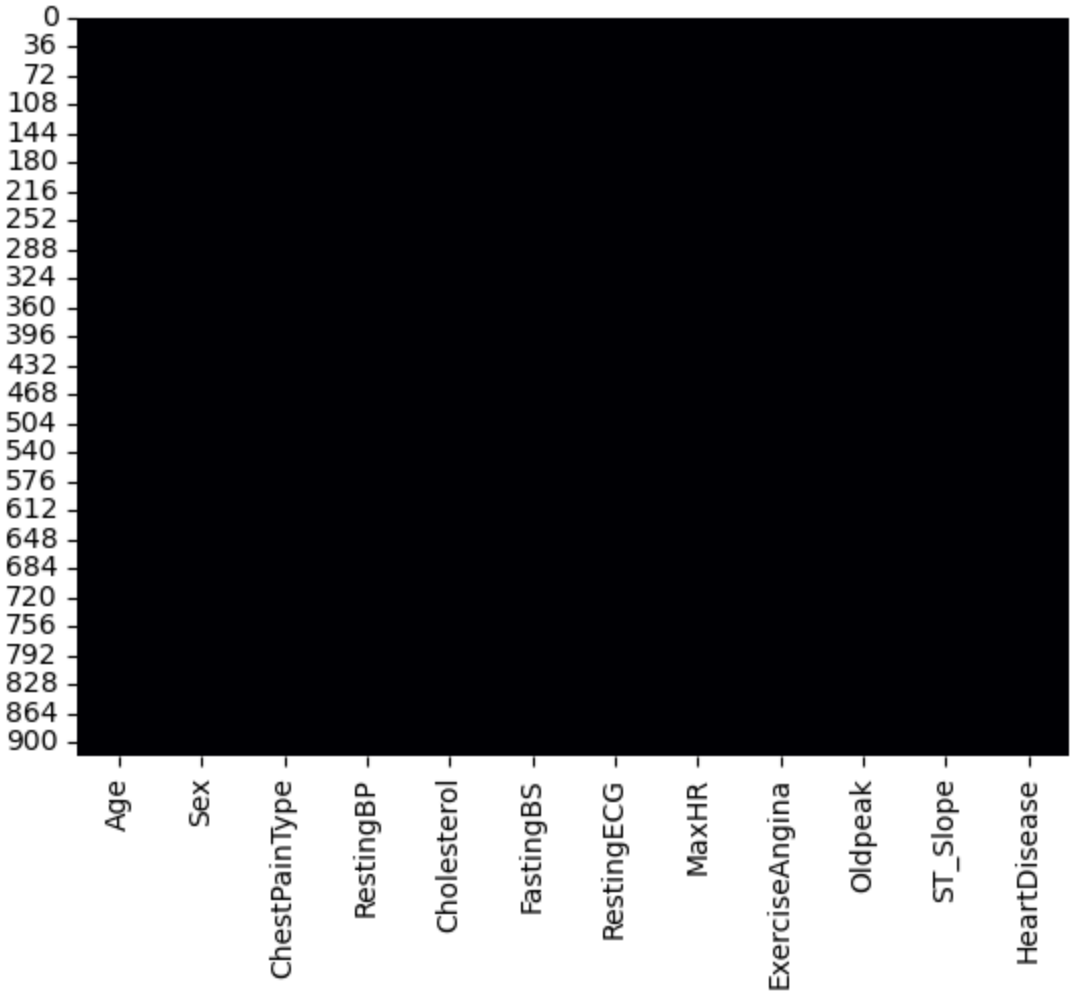
In [4]:
```
1 data.columns
```

Out[4]:
```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingB
S',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
```

In [5]:
```
1 data.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 [6]:
```
1 sns.heatmap(data.isnull(),cmap = 'magma',cbar = False);
```



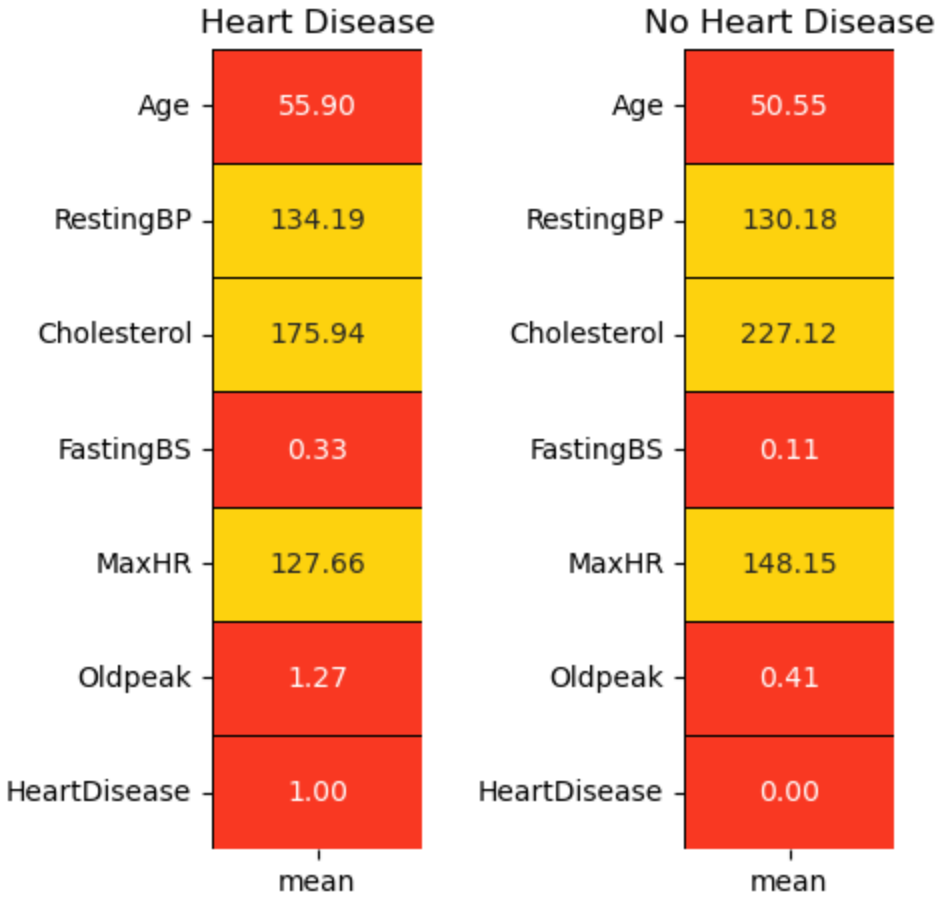
- **No null values** present in the data!

In [7]:
```
1 data.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
Age	918.00	53.51	9.43	28.00	47.00	54.00	60.00	77.00
RestingBP	918.00	132.40	18.51	0.00	120.00	130.00	140.00	200.00
Cholesterol	918.00	198.80	109.38	0.00	173.25	223.00	267.00	603.00
FastingBS	918.00	0.23	0.42	0.00	0.00	0.00	0.00	1.00
MaxHR	918.00	136.81	25.46	60.00	120.00	138.00	156.00	202.00
Oldpeak	918.00	0.89	1.07	-2.60	0.00	0.60	1.50	6.20
HeartDisease	918.00	0.55	0.50	0.00	0.00	1.00	1.00	1.00

In [8]:
```
yes = data[data['HeartDisease'] == 1].describe().T
no = data[data['HeartDisease'] == 0].describe().T
colors = ['#F93822','#FDD20E']

fig,ax = plt.subplots(nrows = 1,ncols = 2,figsize = (5,5))
plt.subplot(1,2,1)
sns.heatmap(yes[['mean']],annot = True,cmap = colors,linewidths = 0.4,lin
plt.title('Heart Disease');

plt.subplot(1,2,2)
sns.heatmap(no[['mean']],annot = True,cmap = colors,linewidths = 0.4,line
plt.title('No Heart Disease');

fig.tight_layout(pad = 2)
```



- **Mean** values of all the features for cases of heart diseases and non-heart diseases.

Exploratory Data Analysis

Dividing features into Numerical and Categorical :

In [9]:
```
col = list(data.columns)
categorical_features = []
numerical_features = []
for i in col:
    if len(data[i].unique()) > 6:
        numerical_features.append(i)
    else:
        categorical_features.append(i)

print('Categorical Features :',*categorical_features)
print('Numerical Features :',*numerical_features)
```

```
Categorical Features : Sex ChestPainType FastingBS RestingECG ExerciseAngina
ST_Slope HeartDisease
Numerical Features : Age RestingBP Cholesterol MaxHR Oldpeak
```

- Here, categorical features are defined if the the attribute has less than 6 unique elements else it is a numerical feature.
- Typical approach for this division of features can also be based on the datatypes of the elements of the respective attribute.

Eg : datatype = integer, attribute = numerical feature ; datatype = string, attribute = categorical feature

- For this dataset, as the number of features are less, we can manually check the dataset as well.

Categorical Features :

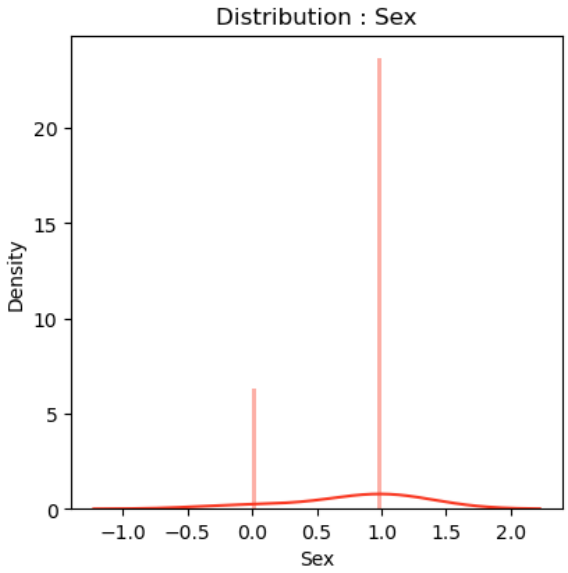
In [10]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df1 = data.copy(deep = True)

df1['Sex'] = le.fit_transform(df1['Sex'])
df1['ChestPainType'] = le.fit_transform(df1['ChestPainType'])
df1['RestingECG'] = le.fit_transform(df1['RestingECG'])
df1['ExerciseAngina'] = le.fit_transform(df1['ExerciseAngina'])
df1['ST_Slope'] = le.fit_transform(df1['ST_Slope'])
```

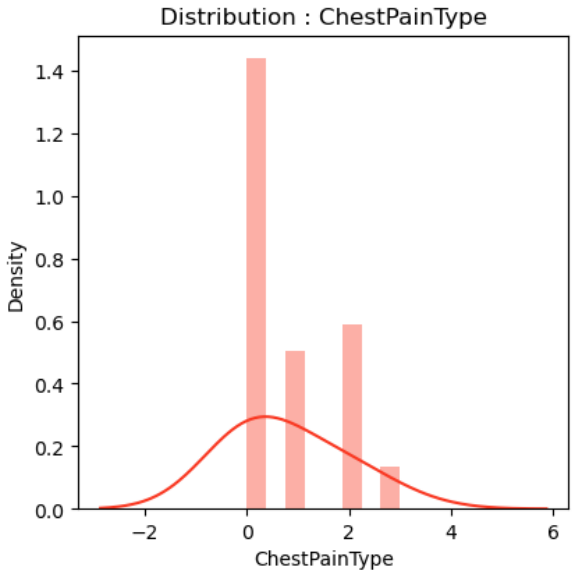
- Creating a deep copy of the orginal dataset and label encoding the text data of the categorical features.
- Modifications in the original dataset will not be highlighted in this deep copy.
- Hence, we use this deep copy of dataset that has all the features converted into numerical values for visualization & modeling purposes.

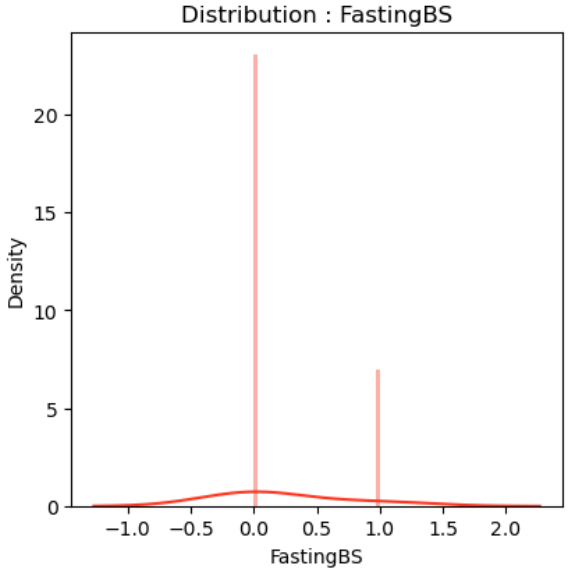
Distribution of Categorical Features :

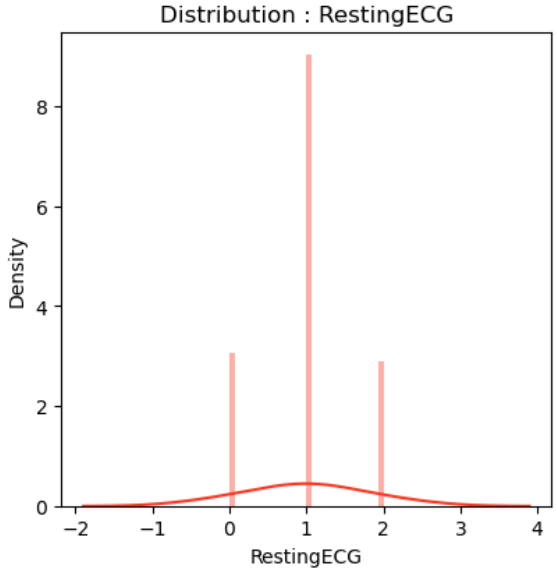
In [11]:

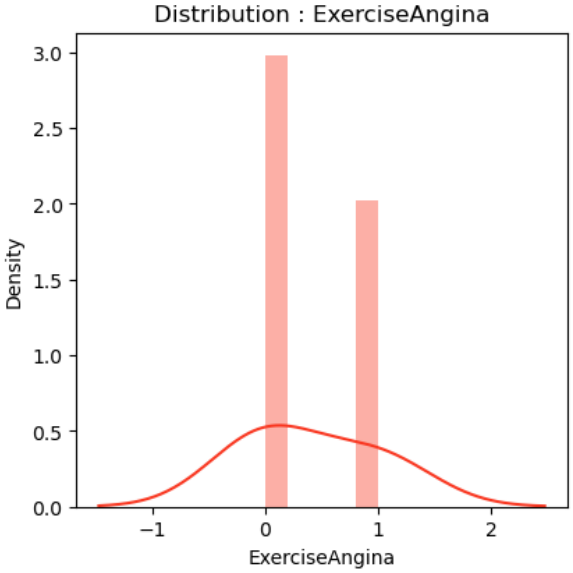
```
fig, ax = plt.subplots(nrows = 3,ncols = 2,figsize = (10,15))
for i in range(len(categorical_features) - 1):

    plt.subplot(3,2,i+1)
    sns.distplot(df1[categorical_features[i]],kde_kws = {'bw' : 1},color
    title = 'Distribution : ' + categorical_features[i]
    plt.title(title)

plt.figure(figsize = (4.75,4.55))
sns.distplot(df1[categorical_features[len(categorical_features) - 1]],kde
title = 'Distribution : ' + categorical_features[len(categorical_features
plt.title(title);
```

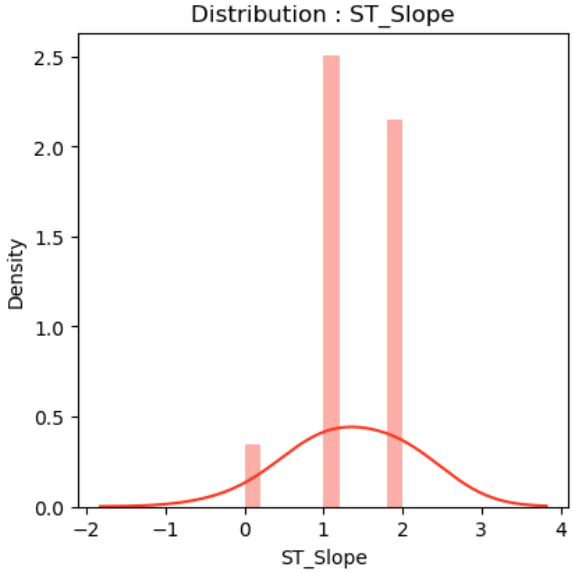

Distribution : Sex
Density
Sex

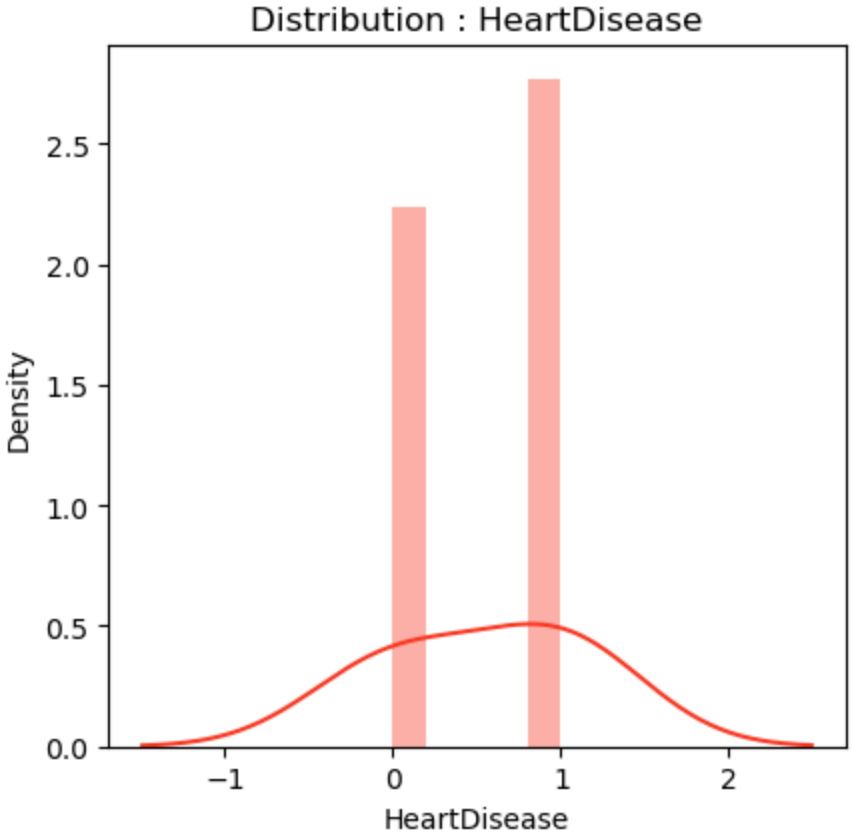

Distribution : ChestPainType
Density
ChestPainType


Distribution : FastingBS
Density
FastingBS


Distribution : RestingECG
Density
RestingECG


Distribution : ExerciseAngina
Density
ExerciseAngina


Distribution : ST_Slope
Density
ST_Slope



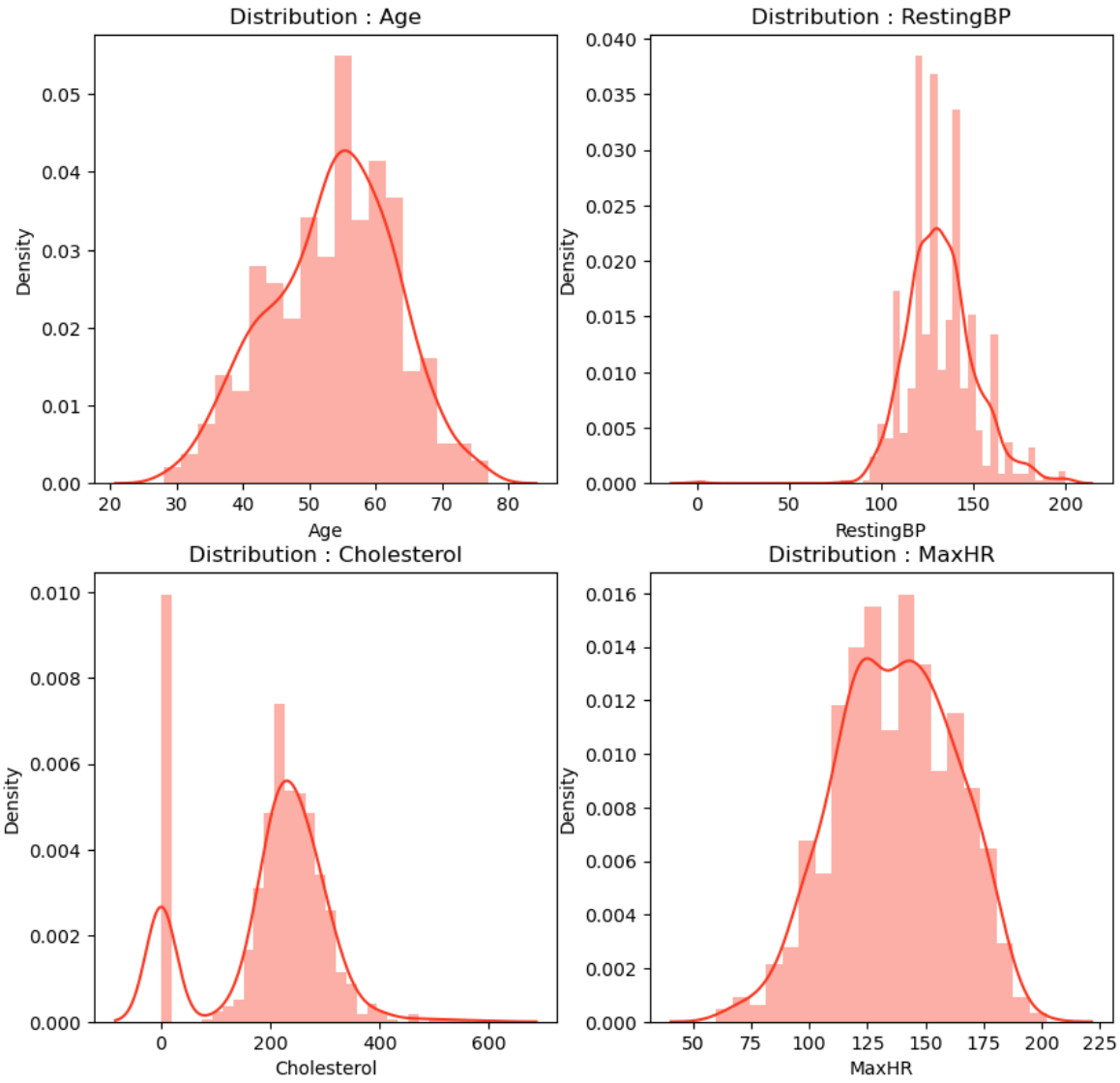
- All the categorical features are near about **Normally Distributed**.

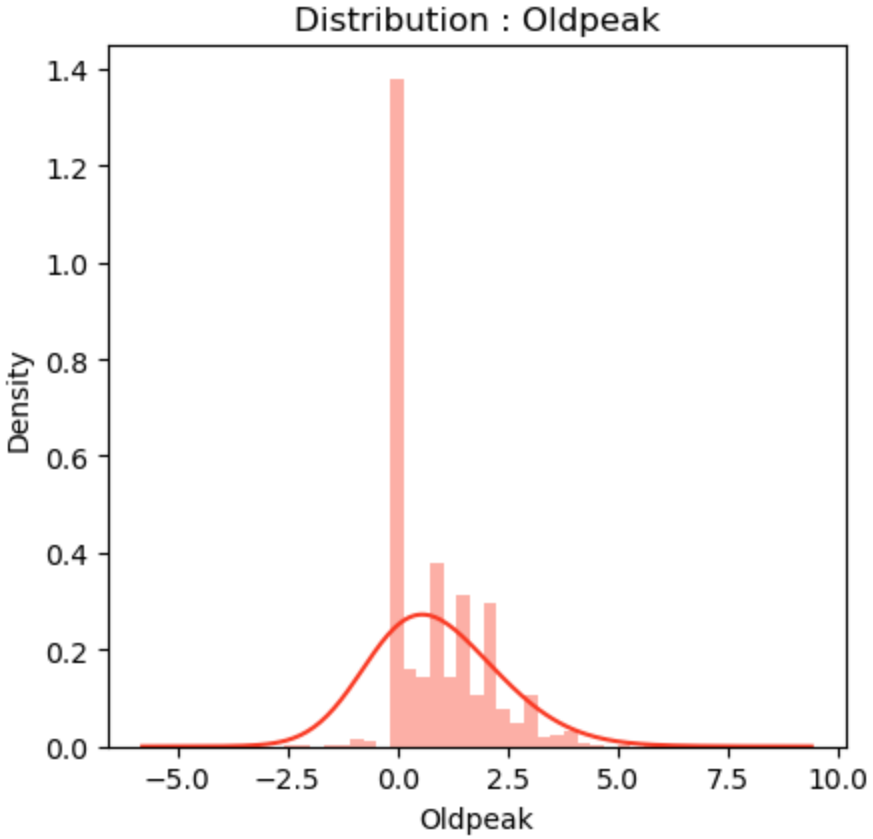
Numerical Features :

Distribution of Numerical Features :

In [12]:

```
fig, ax = plt.subplots(nrows = 2,ncols = 2,figsize = (10,9.75))
for i in range(len(numerical_features) - 1):
    plt.subplot(2,2,i+1)
    sns.distplot(data[numerical_features[i]],color = colors[0])
    title = 'Distribution : ' + numerical_features[i]
    plt.title(title)
plt.show()

plt.figure(figsize = (4.75,4.55))
sns.distplot(df1[numerical_features[len(numerical_features) - 1]],kde_kws
title = 'Distribution : ' + numerical_features[len(numerical_features) -
plt.title(title);
```

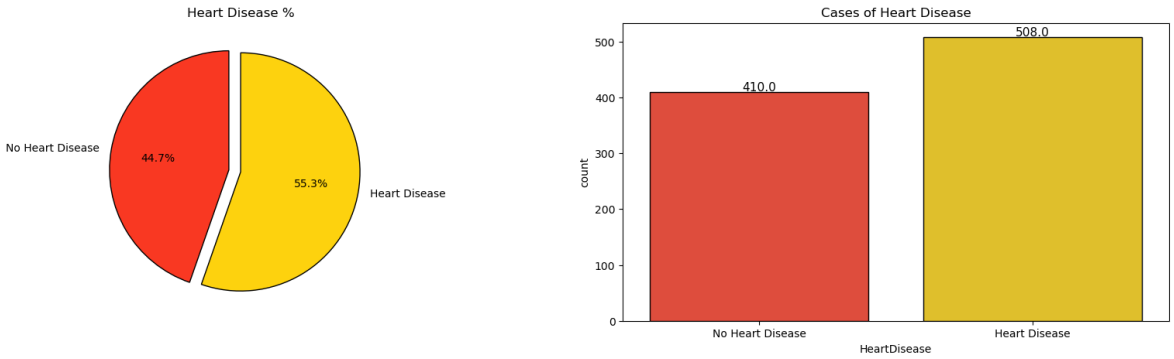




- **Oldpeak's** data distribution is rightly skewed.
- **Cholestrol** has a bidmodal data distribution.

Target Variable Visualization (HeartDisease) :

In [13]:
```
l = list(data['HeartDisease'].value_counts())
circle = [l[1] / sum(l) * 100,l[0] / sum(l) * 100]

fig,ax = plt.subplots(nrows = 1,ncols = 2,figsize = (20,5))
plt.subplot(1,2,1)
plt.pie(circle,labels = ['No Heart Disease','Heart Disease'],autopct='%1.
        wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased'
plt.title('Heart Disease %');

plt.subplot(1,2,2)
ax = sns.countplot(x='HeartDisease', data=data, palette=colors, edgecolor
for rect in ax.patches:
    ax.text(rect.get_x() + rect.get_width() / 2, rect.get_height() + 2, r
ax.set_xticklabels(['No Heart Disease','Heart Disease'])
plt.title('Cases of Heart Disease');
plt.show()
```

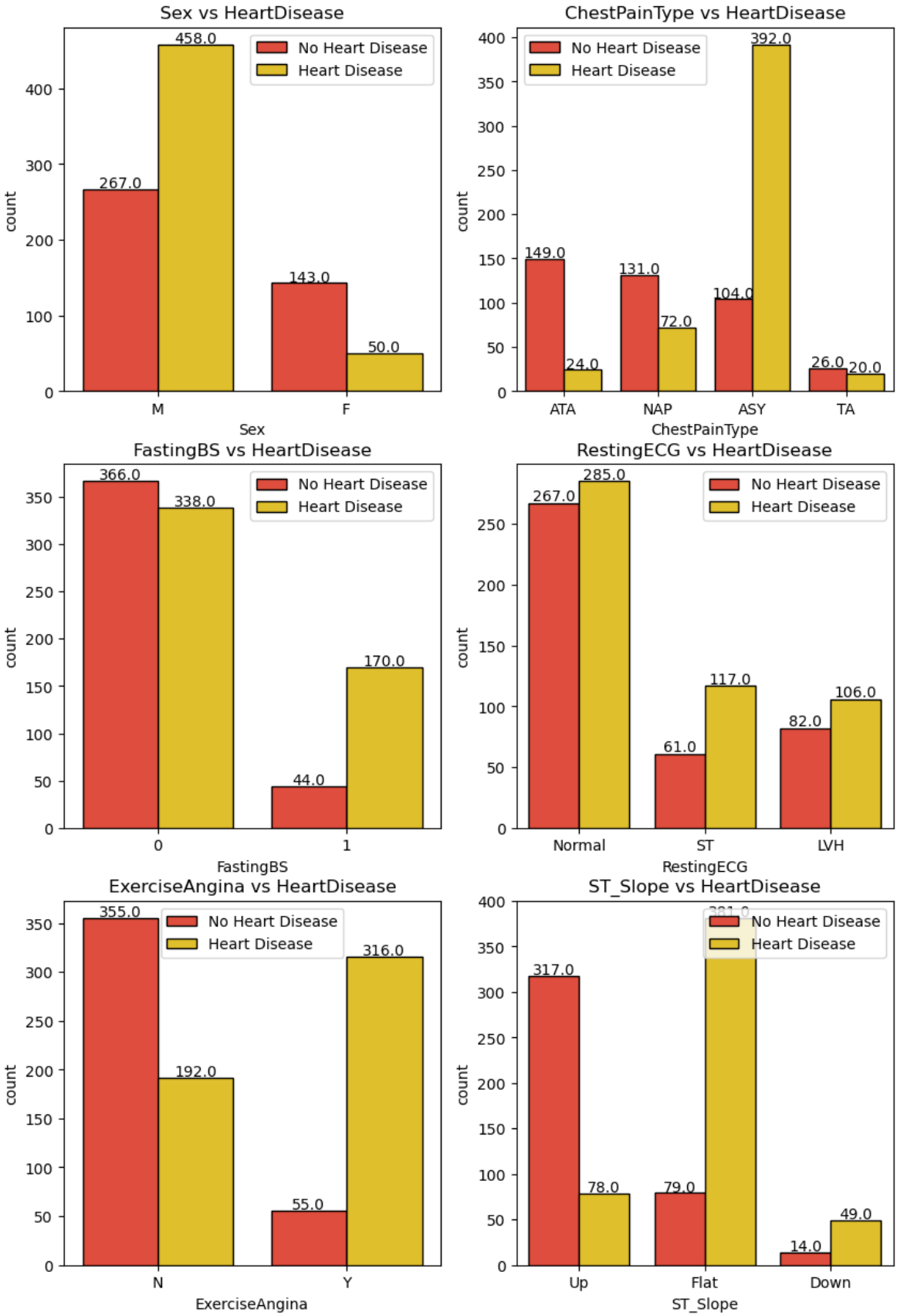


- The dataset is pretty much **evenly balanced!**

Categorical Features vs Target Variable (HeartDisease) :

In [14]:

```
fig, ax = plt.subplots(nrows = 3,ncols = 2,figsize = (10,15))
for i in range(len(categorical_features) - 1):
    plt.subplot(3, 2, i + 1)
    ax = sns.countplot(x=categorical_features[i], data=data, hue="HeartDi
    for rect in ax.patches:
        ax.text(rect.get_x() + rect.get_width() / 2, rect.get_height() +
                horizontalalignment='center', fontsize=10)
    title = categorical_features[i] + ' vs HeartDisease'
    plt.legend(['No Heart Disease','Heart Disease'])
    plt.title(title);
```



- **Male** population has more heart disease patients than no heart disease patients. In the case of **Female** population, heart disease patients are less than no heart disease patients.
- **ASY** type of chest pain boldly points towards major chances of heart disease.

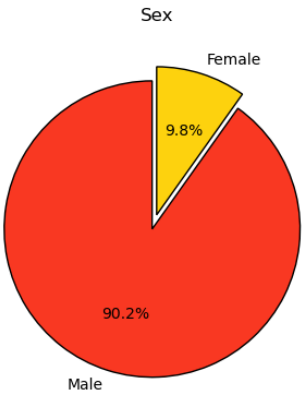
- **Fasting Blood Sugar** is tricky! Patients diagnosed with Fasting Blood Sugar and no Fasting Blood Sugar have significant heart disease patients.
- **RestingECG** does not present with a clear cut category that highlights heart disease patients. All the 3 values consist of high number of heart disease patients.
- **Exercise Induced Engina** definitely bumps the probability of being diagnosed with heart diseases.
- With the **ST_Slope** values, **flat** slope displays a very high probability of being diagnosed

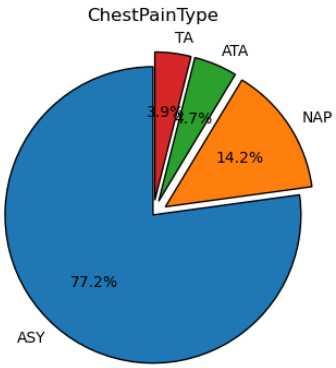
Categorical Features vs Positive Heart Disease Cases :

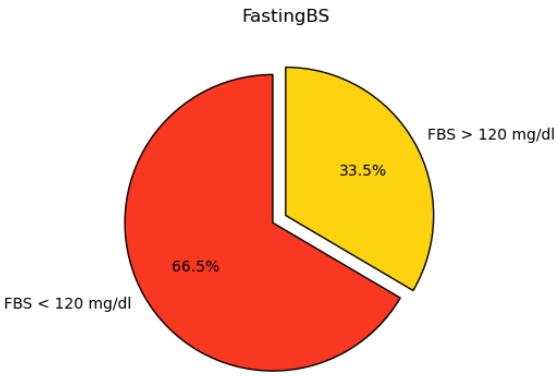
In [15]:
```
sex = data[data['HeartDisease'] == 1]['Sex'].value_counts()
sex = [sex[0] / sum(sex) * 100, sex[1] / sum(sex) * 100]

cp = data[data['HeartDisease'] == 1]['ChestPainType'].value_counts()
cp = [cp[0] / sum(cp) * 100,cp[1] / sum(cp) * 100,cp[2] / sum(cp) * 100,c

fbs = data[data['HeartDisease'] == 1]['FastingBS'].value_counts()
fbs = [fbs[0] / sum(fbs) * 100,fbs[1] / sum(fbs) * 100]

restecg = data[data['HeartDisease'] == 1]['RestingECG'].value_counts()
restecg = [restecg[0] / sum(restecg) * 100,restecg[1] / sum(restecg) * 10

exang = data[data['HeartDisease'] == 1]['ExerciseAngina'].value_counts()
exang = [exang[0] / sum(exang) * 100,exang[1] / sum(exang) * 100]

slope = data[data['HeartDisease'] == 1]['ST_Slope'].value_counts()
slope = [slope[0] / sum(slope) * 100,slope[1] / sum(slope) * 100,slope[2]
```

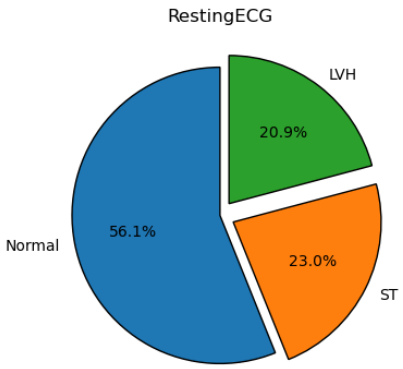
In [16]:

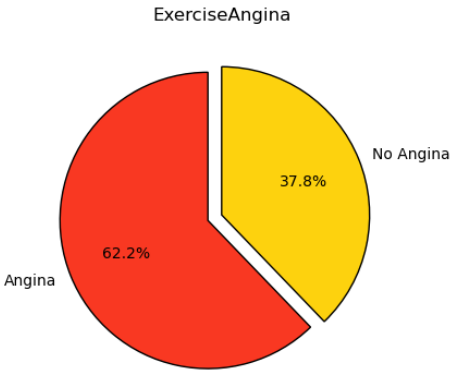
```
ax,fig = plt.subplots(nrows = 4,ncols = 2,figsize = (15,15))

plt.subplot(3,2,1)
plt.pie(sex,labels = ['Male','Female'],autopct='%1.1f%%',startangle = 90,
        wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased'
plt.title('Sex');

plt.subplot(3,2,2)
plt.pie(cp,labels = ['ASY', 'NAP', 'ATA', 'TA'],autopct='%1.1f%%',startan
        wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased'
plt.title('ChestPainType');

plt.subplot(3,2,3)
plt.pie(fbs,labels = ['FBS < 120 mg/dl','FBS > 120 mg/dl'],autopct='%1.1f
        wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased'
plt.title('FastingBS');

plt.subplot(3,2,4)
plt.pie(restecg,labels = ['Normal','ST','LVH'],autopct='%1.1f%%',startang
        wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased'
plt.title('RestingECG');

plt.subplot(3,2,5)
plt.pie(exang,labels = ['Angina','No Angina'],autopct='%1.1f%%',startangl
        wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased'
plt.title('ExerciseAngina');

plt.subplot(3,2,6)
plt.pie(slope,labels = ['Flat','Up','Down'],autopct='%1.1f%%',startangle
        wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased'
plt.title('ST_Slope');
```

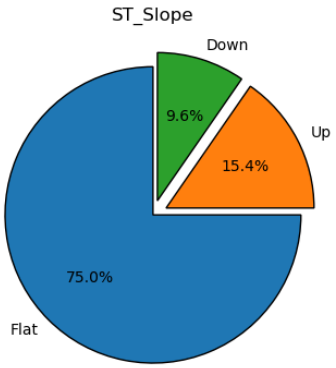










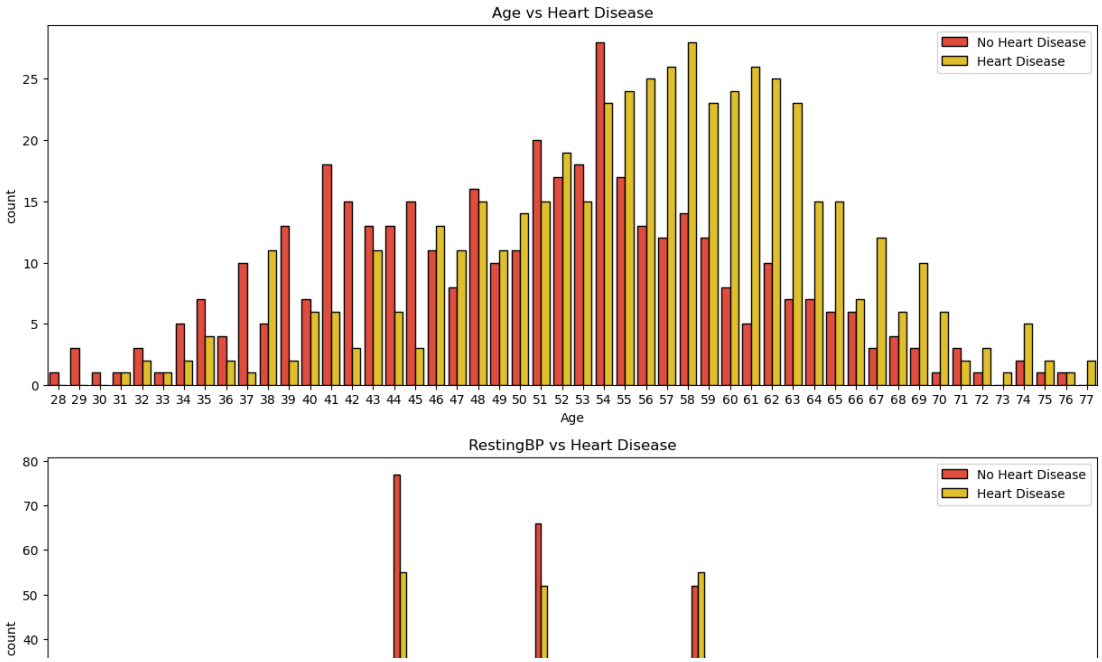


- Out of all the heart disease patients, a staggering 90% patients are **male**.
- When it comes to the type of chest pain, **ASY** type holds the majority with 77% that lead to heart diseases.
- **Fasting Blood Sugar** level < 120 mg/dl displays high chances of heart diseases.
- For **RestingECG**, **Normal** level accounts for 56% chances of heart diseases than **LVH** and **ST** levels.
- Detection of **Exercise Induced Angina** also points towards heart diseases.
- When it comes to **ST_Slope** readings, **Flat** level holds a massive chunk with 75% that may assist in detecting underlying heart problems.

Numerical Features vs Target Variable (HeartDisease) :

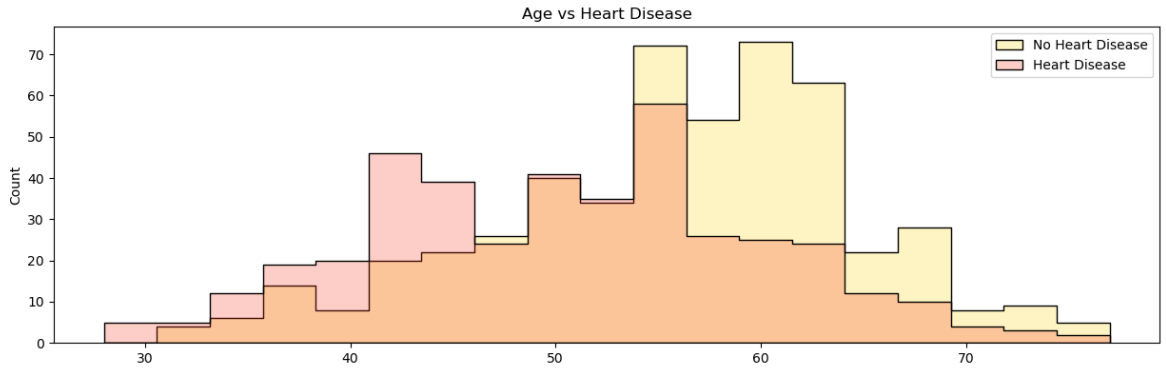
In [17]:

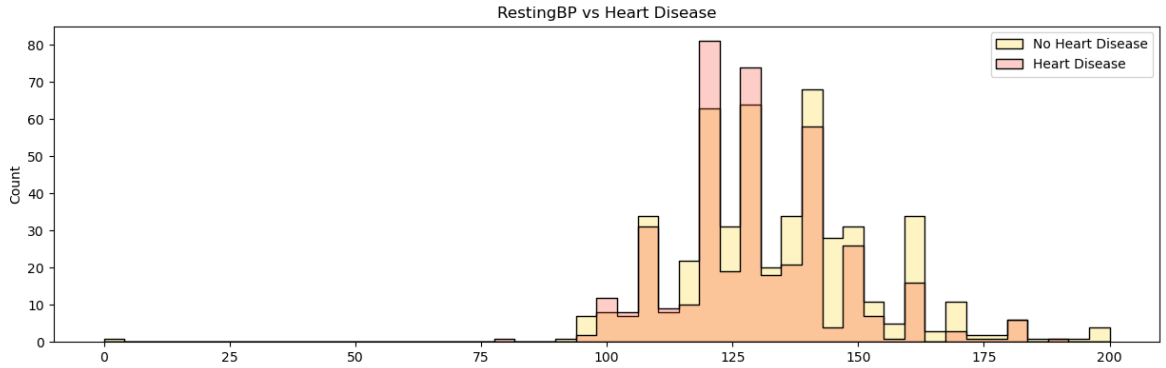
```
fig, ax = plt.subplots(nrows = 5,ncols = 1,figsize = (15,30))
for i in range(len(numerical_features)):
    plt.subplot(5,1,i+1)
    sns.countplot(x = numerical_features[i],data = data,hue = "HeartDisea
    title = numerical_features[i] + ' vs Heart Disease'
    plt.legend(['No Heart Disease','Heart Disease'])
    plt.title(title);
```

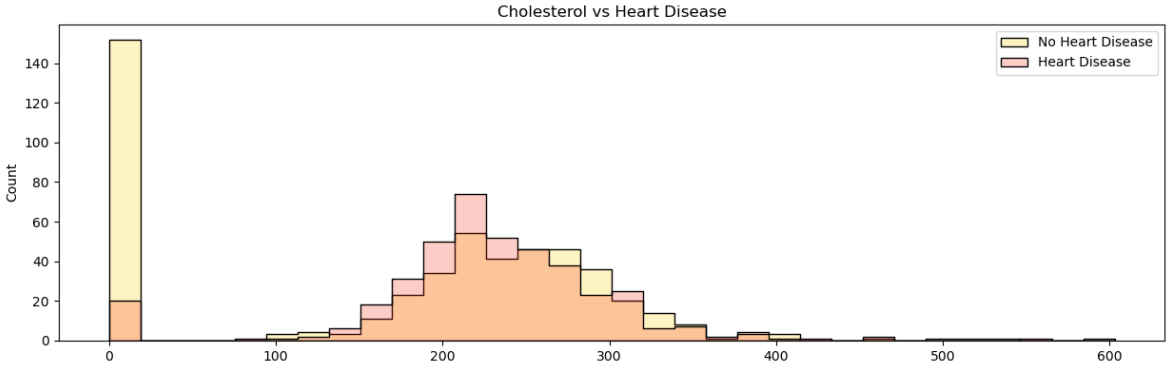


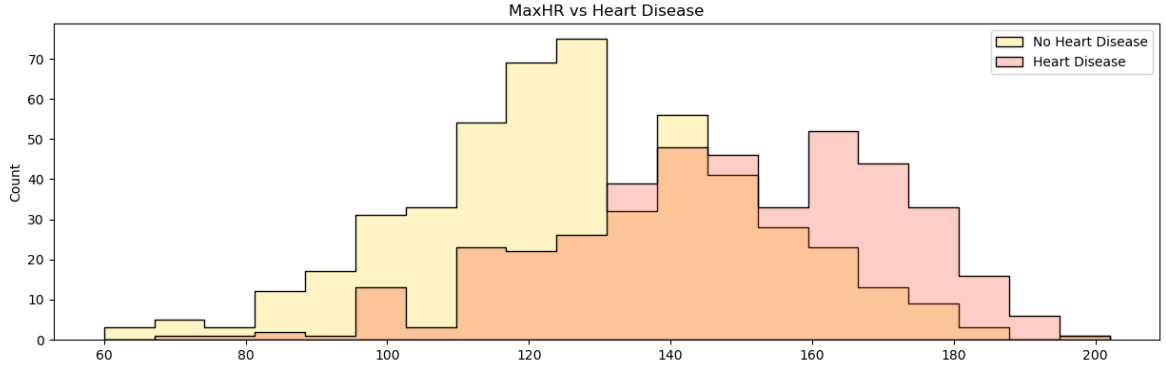
In [18]:

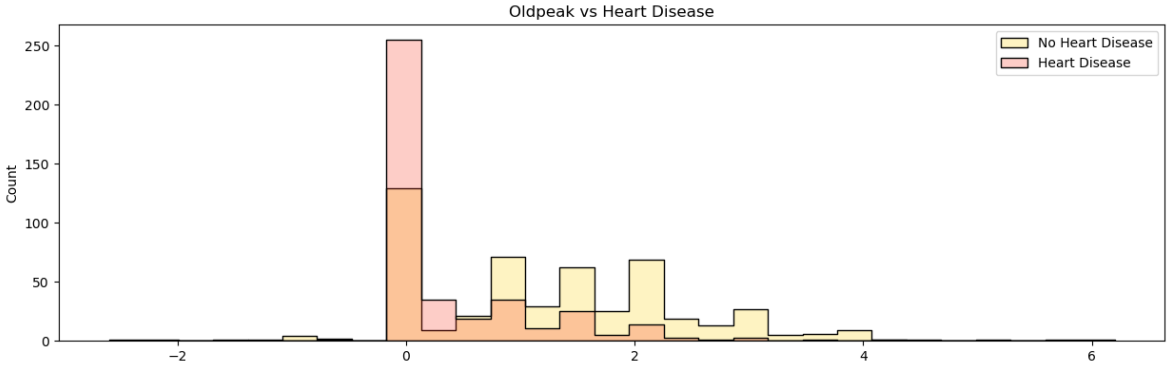
```
fig, ax = plt.subplots(nrows = 5,ncols = 1,figsize = (15,25))
for i in range(len(numerical_features)):
    plt.subplot(5, 1, i + 1)
    sns.histplot(data, x=numerical_features[i], hue="HeartDisease", eleme
    title = numerical_features[i] + ' vs Heart Disease'
    plt.legend(['No Heart Disease', 'Heart Disease'])
    plt.title(title)
    plt.xlabel('')
    plt.ylabel('Count')

```


Age vs Heart Disease
No Heart Disease
Heart Disease
Count
0
10
20
30
40
50
60
70
30
40
50
60
70


RestingBP vs Heart Disease
No Heart Disease
Heart Disease
Count
0
10
20
30
40
50
60
70
80
0
25
50
75
100
125
150
175
200


Cholesterol vs Heart Disease
No Heart Disease
Heart Disease
Count
0
20
40
60
80
100
120
140
0
100
200
300
400
500
600

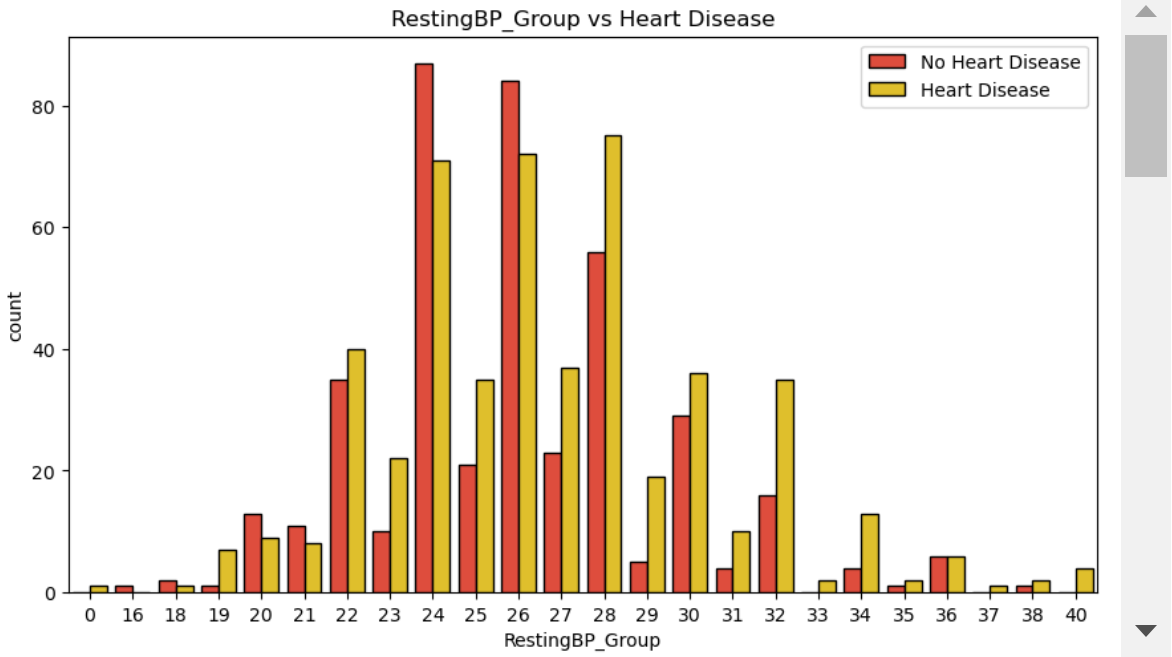

MaxHR vs Heart Disease
No Heart Disease
Heart Disease
Count
0
10
20
30
40
50
60
70
60
80
100
120
140
160
180
200


Oldpeak vs Heart Disease
No Heart Disease
Heart Disease
Count
0
50
100
150
200
250
−2
0
2
4
6

- Because of too many unique data points in the above features, it is difficult to gain any type of insight. Thus, we will convert these numerical features,except age, into categorical features for understandable visualization and gaining insights purposes.
- Thus, we scale the individual values of these features. This brings the varied data points to a constant value that represents a range of values.
- Here, we divide the data points of the numerical features by 5 or 10 and assign its quotient value as the representative constant for that data point. The scaling constants of 5 & 10 are decided by looking into the data & intuition.

In [19]:
```
data['RestingBP_Group'] = [ int(i / 5) for i in data['RestingBP']]
data['Cholesterol_Group'] = [ int(i / 10) for i in data['Cholesterol']]
data['MaxHR_Group'] = [ int(i / 5) for i in data['MaxHR']]
data['Oldpeak_Group'] = [ int( (i*10) / 5) for i in data['Oldpeak']]
```

In [20]:
```
fig, ax = plt.subplots(nrows = 4,ncols = 1,figsize = (10,25))
group_numerical_features = [i + '_Group' for i in numerical_features[1:]]
for i in range(len(group_numerical_features)):
    plt.subplot(4,1,i+1)
    sns.countplot(x = group_numerical_features[i],data = data,hue = "Hear
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = group_numerical_features[i] + ' vs Heart Disease'
    plt.title(title);
```

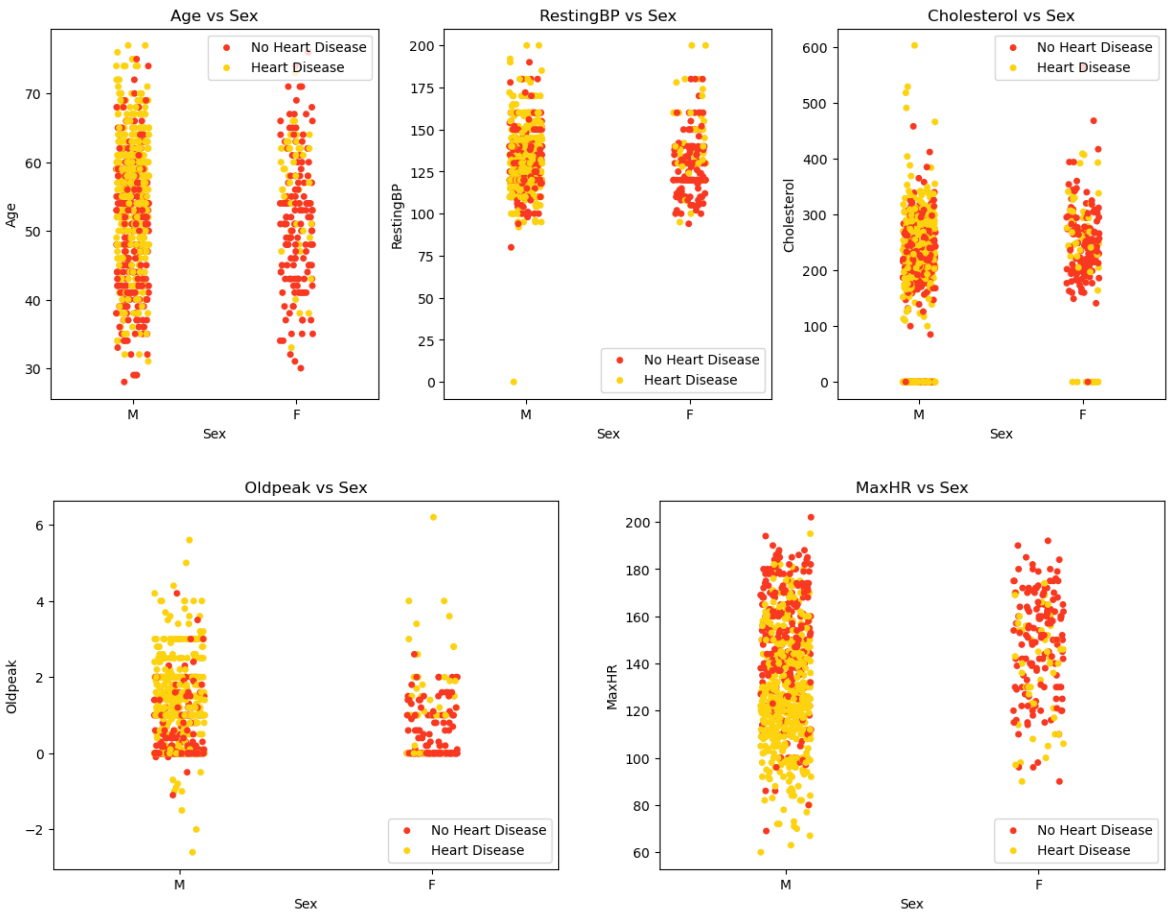


- From the **RestingBP** group data, **95** (19x5) - **170** (34x5) readings are most prone to be detected with heart diseases.
- **Cholesterol** levels between **160** (16x10) - **340** (34x10) are highly susceptible to heart diseases.
- For the **MaxHR** readings, heart diseases are found throughout the data but **70** (14x5) - **180** (36x5) values has detected many cases.
- **Oldpeak** values also display heart diseases throughout. **0** (0x5/10) - **4** (8x5/10) slope values display high probability to be diagnosed with heart diseases.

Numerical features vs Categorical features w.r.t Target variable(HeartDisease) :

Sex vs Numerical Features :

In [21]:
```
fig,ax = plt.subplots(nrows = 1,ncols = 3,figsize = (15,5))
for i in range(3):
    plt.subplot(1,3,i+1)
    sns.stripplot(x = 'Sex',y = numerical_features[i],data = data,hue = '
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs Sex'
    plt.title(title);

fig,ax = plt.subplots(nrows = 1,ncols = 2,figsize = (15,5))
for i in [-1,-2]:
    plt.subplot(1,2,-i)
    sns.stripplot(x = 'Sex',y = numerical_features[i],data = data,hue = '
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs Sex'
    plt.title(title);
```

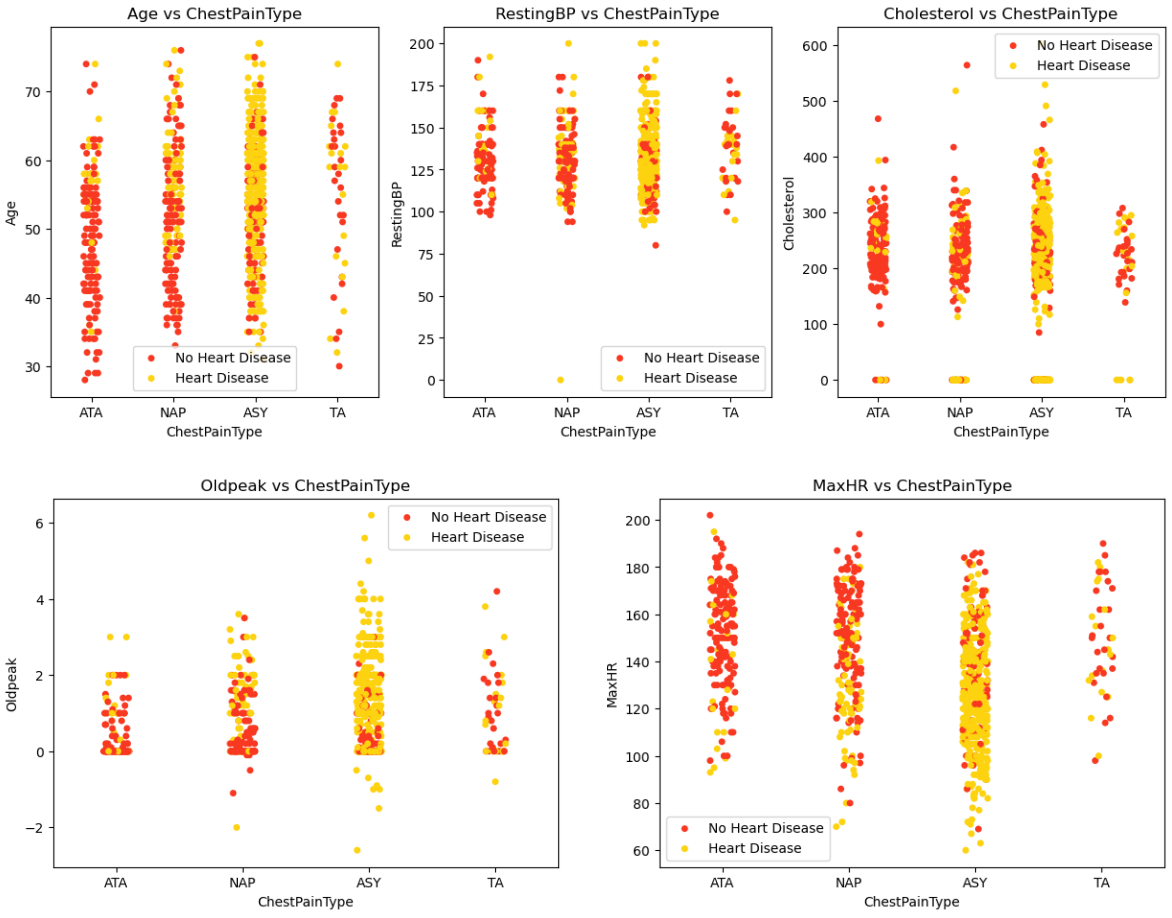


- **Male** population displays heart diseases at near about all the values of the numerical features. Above the age of 50, positive old peak values and maximum heart rate below 140, heart diseases in male population become dense.

- **Female** population data points are very less as compared to **male** population data points. Hence, we cannot point to specific ranges or values that display cases of heart diseases.

ChestPainType vs Numerical Features :

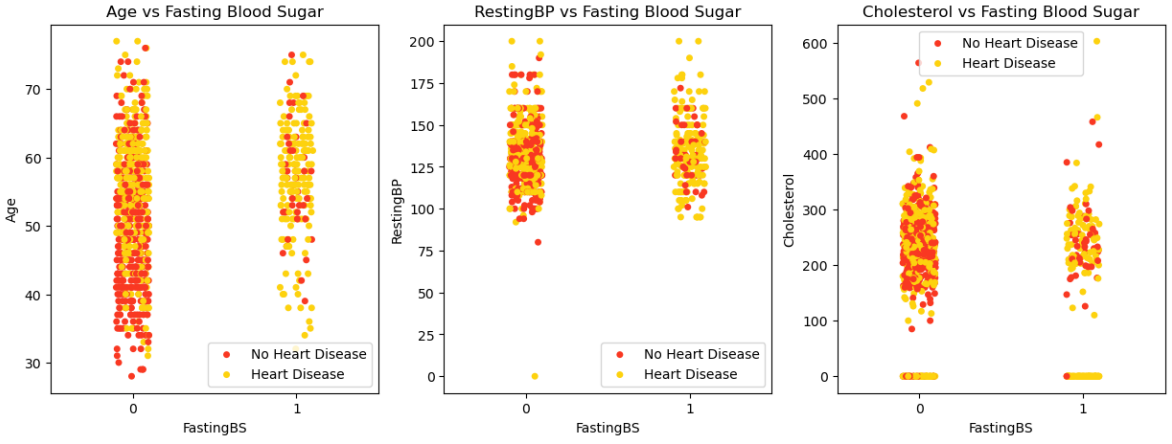
In [22]:
```
fig,ax = plt.subplots(nrows = 1,ncols = 3,figsize = (15,5))
for i in range(3):
    plt.subplot(1,3,i+1)
    sns.stripplot(x = 'ChestPainType',y = numerical_features[i],data = da
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs ChestPainType'
    plt.title(title);

fig,ax = plt.subplots(nrows = 1,ncols = 2,figsize = (15,5))
for i in [-1,-2]:
    plt.subplot(1,2,-i)
    sns.stripplot(x = 'ChestPainType',y = numerical_features[i],data = da
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs ChestPainType'
    plt.title(title);
```

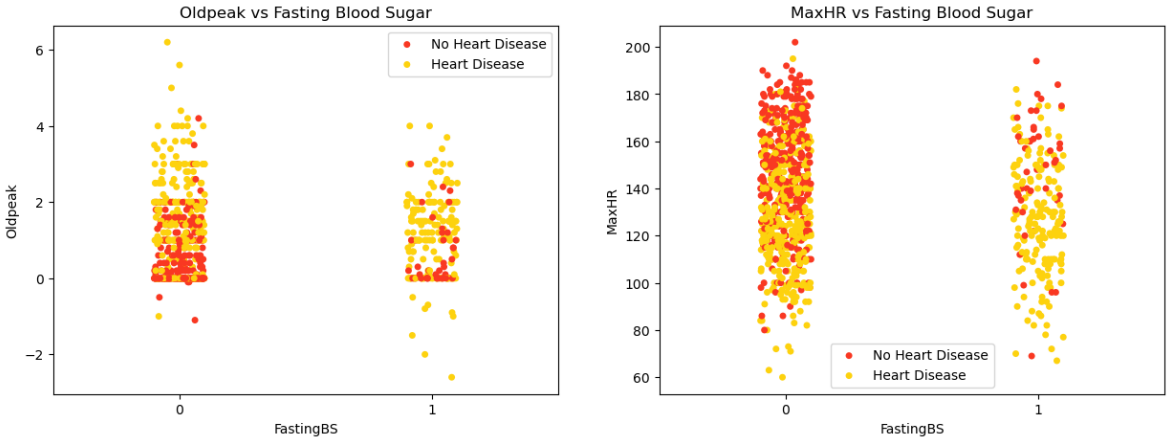


- **ASY** type of chest pain dominates other types of chest pain in all the numerical features by a lot.

FastingBS vs Numerical features :

In [23]:
```
fig,ax = plt.subplots(nrows = 1,ncols = 3,figsize = (15,5))
for i in range(3):
    plt.subplot(1,3,i+1)
    sns.stripplot(x = 'FastingBS',y = numerical_features[i],data = data,h
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs Fasting Blood Sugar'
    plt.title(title);

fig,ax = plt.subplots(nrows = 1,ncols = 2,figsize = (15,5))
for i in [-1,-2]:
    plt.subplot(1,2,-i)
    sns.stripplot(x = 'FastingBS',y = numerical_features[i],data = data,h
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs Fasting Blood Sugar'
    plt.title(title);
```



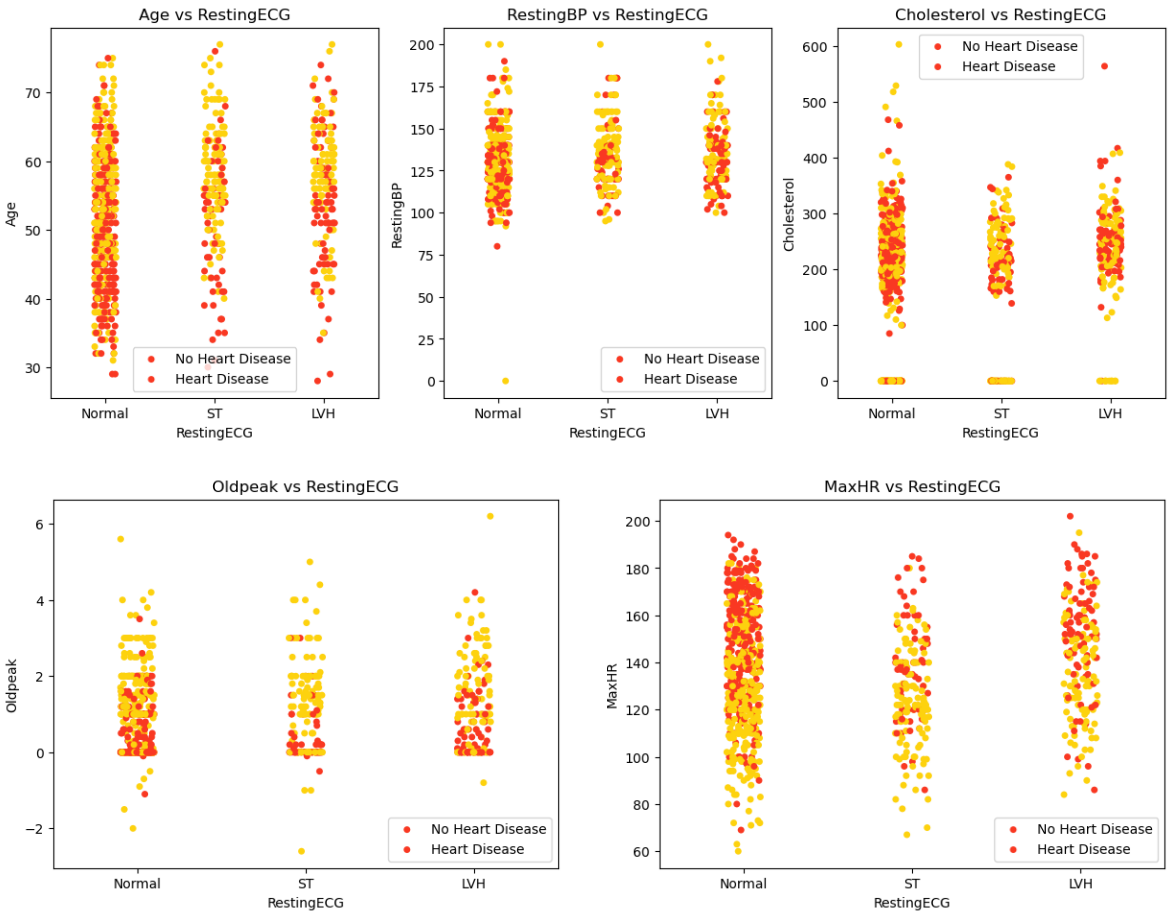


- Above the **age** 50, heart diseases are found throughout the data irrespective of the patient being diagnosed with Fasting Blood Sugar or not.
- **Fasting Blood Sugar** with **Resting BP** over 100 has displayed more cases of heart diseases than patients with no fasting blood sugar.
- **Cholesterol** with **Fasting Blood Sugar** does not seem to have an effect in understanding reason behind heart diseases.

- Patients that have not been found positive with **Fasting Blood Sugar** but have maximum

RestingECG vs Numerical Features :

In [24]:
```
fig,ax = plt.subplots(nrows = 1,ncols = 3,figsize = (15,5))
for i in range(3):
    plt.subplot(1,3,i+1)
    sns.stripplot(x = 'RestingECG',y = numerical_features[i],data = data,
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs RestingECG'
    plt.title(title);

fig,ax = plt.subplots(nrows = 1,ncols = 2,figsize = (15,5))
for i in [-1,-2]:
    plt.subplot(1,2,-i)
    sns.stripplot(x = 'RestingECG',y = numerical_features[i],data = data,
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs RestingECG'
    plt.title(title);
```

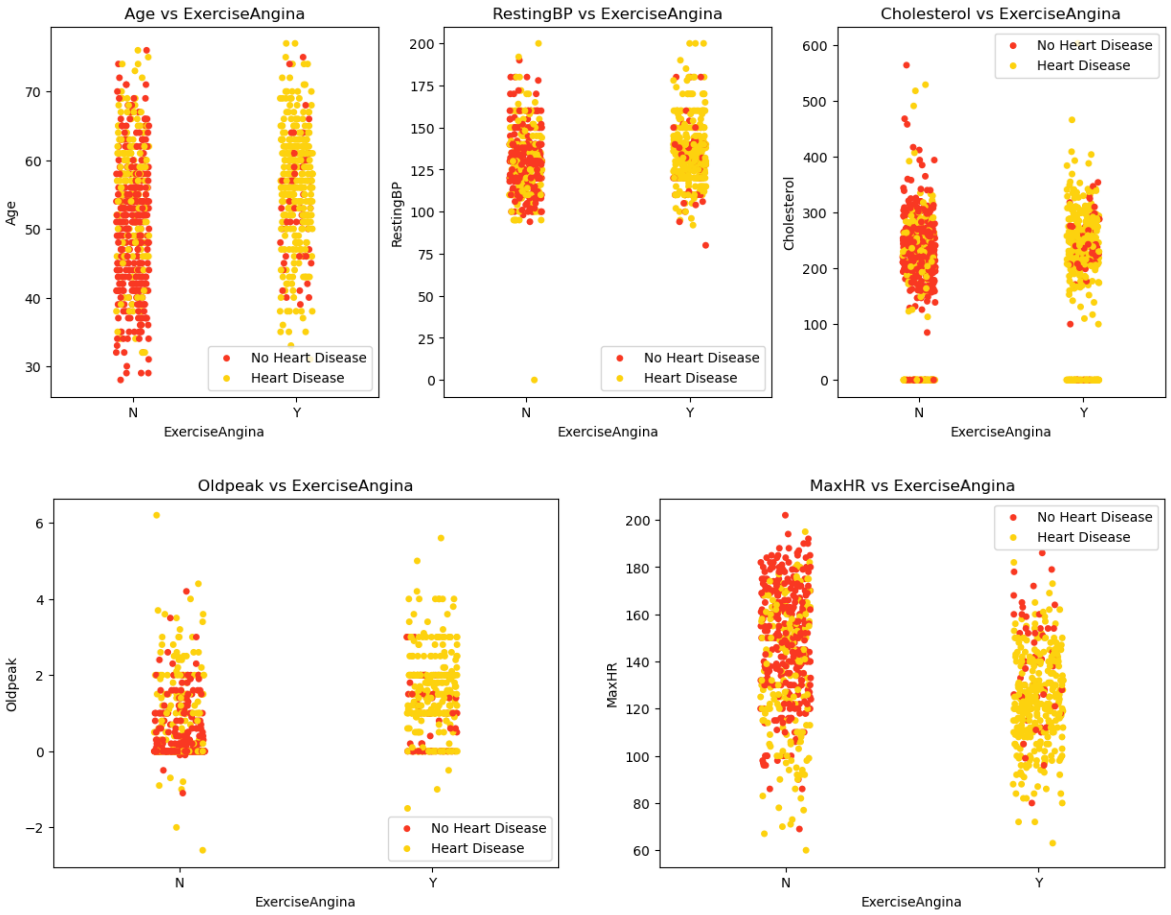


- Heart diseases with **RestingECG** values of **Normal**, **ST** and **LVH** are detected starting from 30,40 & 40 respectively. Patients above the age of 50 are more prone than anyother ages irrespective of **RestingECG** values.
- Heart diseases are found consistently throughout any values of **RestingBP** and **RestingECG**.

- **Cholesterol** values between 200 - 300 coupled with **ST** value of **RestingECG** display a patch of patients suffering from heart diseases.
- For **maximum Heart Rate** values, heart diseases are detected in dense below 140 points and **Normal** RestingECG. **ST** & **LVH** throughout the maximum heart rate values display heart disease cases.

ExerciseAngina vs Numerical Features :

In [25]:
```
fig,ax = plt.subplots(nrows = 1,ncols = 3,figsize = (15,5))
for i in range(3):
    plt.subplot(1,3,i+1)
    sns.stripplot(x = 'ExerciseAngina',y = numerical_features[i],data = d
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs ExerciseAngina'
    plt.title(title);

fig,ax = plt.subplots(nrows = 1,ncols = 2,figsize = (15,5))
for i in [-1,-2]:
    plt.subplot(1,2,-i)
    sns.stripplot(x = 'ExerciseAngina',y = numerical_features[i],data = d
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs ExerciseAngina'
    plt.title(title);
```

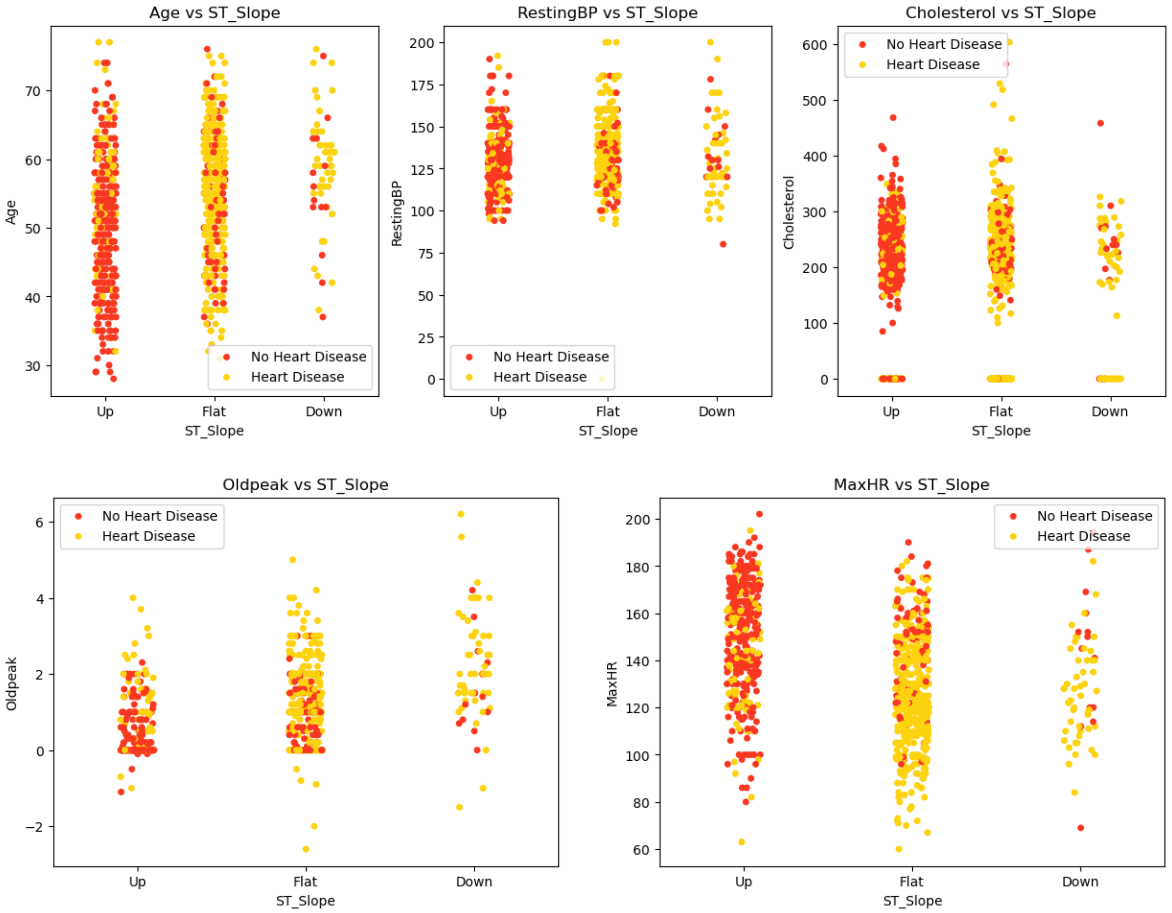


- A crsytal clear observation can be made about the relationship between **heart disease** case and **Exercise induced Angina**. A positive correlation between the 2 features can be

concluded throughout all the numerical features.

ST_Slope vs Numerical Features :

In [26]:
```
fig,ax = plt.subplots(nrows = 1,ncols = 3,figsize = (15,5))
for i in range(3):
    plt.subplot(1,3,i+1)
    sns.stripplot(x = 'ST_Slope',y = numerical_features[i],data = data,hu
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs ST_Slope'
    plt.title(title);

fig,ax = plt.subplots(nrows = 1,ncols = 2,figsize = (15,5))
for i in [-1,-2]:
    plt.subplot(1,2,-i)
    sns.stripplot(x = 'ST_Slope',y = numerical_features[i],data = data,hu
    plt.legend(['No Heart Disease', 'Heart Disease'])
    title = numerical_features[i] + ' vs ST_Slope'
    plt.title(title);
```

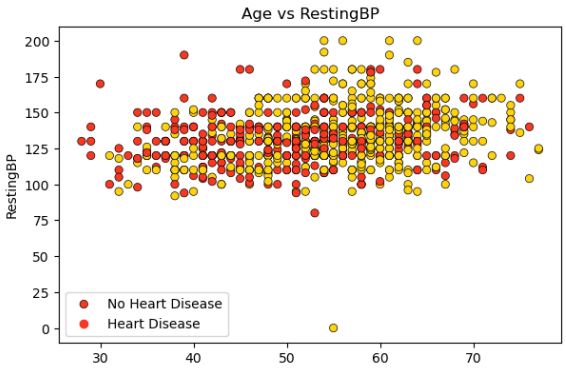


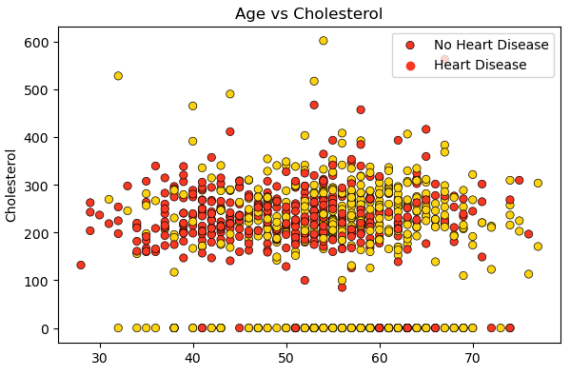
- Another crystal clear positive observation can be made about the positive correlation between **ST_Slope** value and **Heart Disease** cases.
- **Flat**, **Down** and **Up** in that order display high, middle and low probability of being diagnosed with heart diseases respectively.

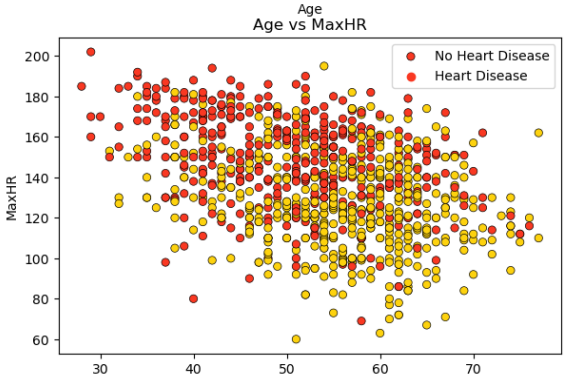
Numerical features vs Numerical features w.r.t Target variable(HeartDisease) :

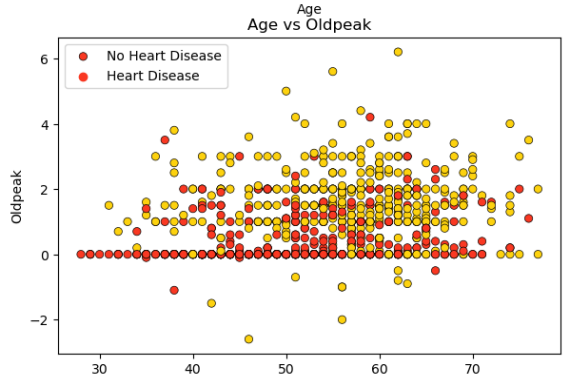
In [27]:

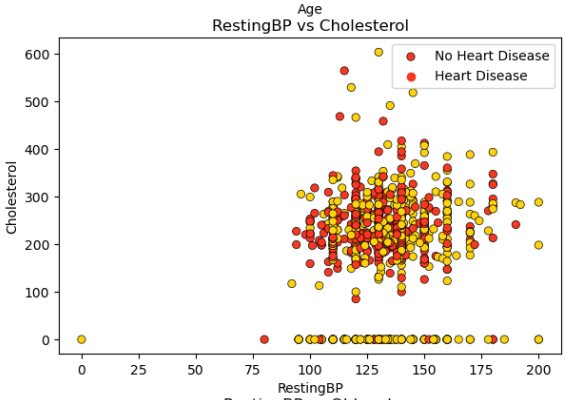
```
a = 0
fig,ax = plt.subplots(nrows = 5,ncols = 2,figsize = (15,25))
for i in range(len(numerical_features)):
    for j in range(len(numerical_features)):
        if i != j and j > i:
            a += 1
            plt.subplot(5,2,a)
            sns.scatterplot(x = numerical_features[i],y = numerical_featu
            plt.legend(['No Heart Disease', 'Heart Disease'])
            title = numerical_features[i] + ' vs ' + numerical_features[j
            plt.title(title)
```

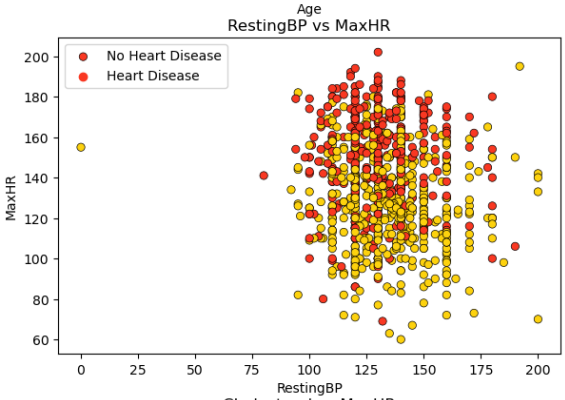

Age vs RestingBP
RestingBP
Age
No Heart Disease
Heart Disease

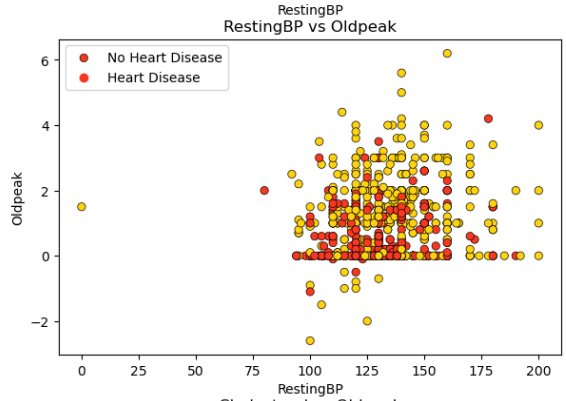

Age vs Cholesterol
Cholesterol
Age
No Heart Disease
Heart Disease

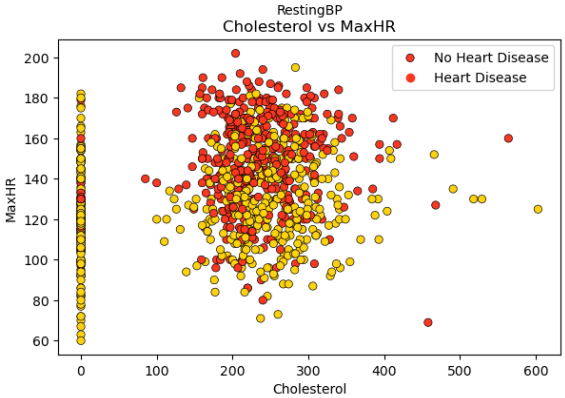

Age vs MaxHR
MaxHR
Age
No Heart Disease
Heart Disease

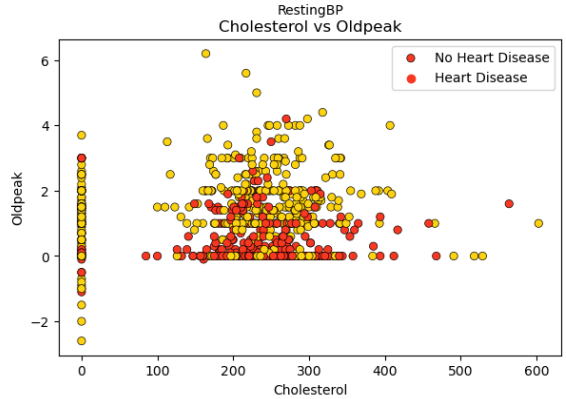

Age vs Oldpeak
Oldpeak
Age
No Heart Disease
Heart Disease

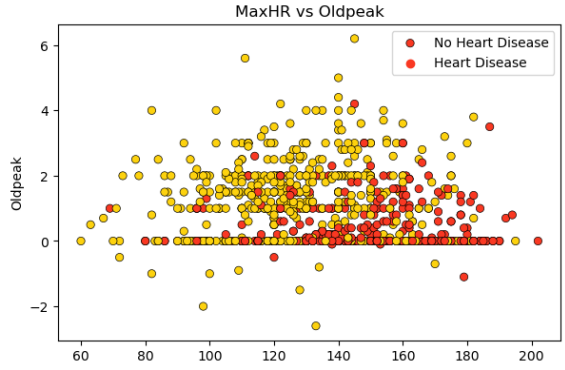

RestingBP vs Cholesterol
Cholesterol
RestingBP
No Heart Disease
Heart Disease


RestingBP vs MaxHR
MaxHR
RestingBP
No Heart Disease
Heart Disease


RestingBP vs Oldpeak
Oldpeak
RestingBP
No Heart Disease
Heart Disease


Cholesterol vs MaxHR
MaxHR
Cholesterol
No Heart Disease
Heart Disease


Cholesterol vs Oldpeak
Oldpeak
Cholesterol
No Heart Disease
Heart Disease


MaxHR vs Oldpeak
Oldpeak
MaxHR
No Heart Disease
Heart Disease

- For **age** 50+, **RestingBP** between 100 - 175, **Cholesterol** level of 200 - 300,**Max Heart Rate** below 160 and positive **oldpeak** values displays high cases of heart disease.
- For **RestingBP** values 100 - 175, highlights too many heart disease patients for all the features.
- **Cholesterol** values 200 - 300 dominates the heart disease cases.
- Similarly, **Max Heart Rate** values below 140 has high probability of being diagnosed with heart diseases.

Summary of EDA

Order / Values of features for positive cases of heart disease :

- **Categorical Features (Order) :**
  - Sex : Male > Female
  - ChestPainType : ASY > NAP > ATA > TA
  - FastingBS : (FBS < 120 mg/dl) > (FBS > 120 mg/dl)
  - RestingECG : Normal > ST > LVH
  - ExerciseAngina : Angina > No Angina
  - ST_Slope : Flat > Up > Down
- **Numerical Features (Range) :**
  - Age : 50+
  - RestingBP : 95 - 170
  - Cholesterol : 160 - 340
  - MaxHR : 70 - 180
  - Oldpeak : 0 - 4

Now that we have understood the typical values of the features, we move on to the next step where we select the appropriate features for modeling!

Feature Engineering

Data Scaling :

In [28]:
```
from sklearn.preprocessing import MinMaxScaler,StandardScaler
mms = MinMaxScaler() # Normalization
ss = StandardScaler() # Standardization

df1['Oldpeak'] = mms.fit_transform(df1[['Oldpeak']])
df1['Age'] = ss.fit_transform(df1[['Age']])
df1['RestingBP'] = ss.fit_transform(df1[['RestingBP']])
df1['Cholesterol'] = ss.fit_transform(df1[['Cholesterol']])
df1['MaxHR'] = ss.fit_transform(df1[['MaxHR']])
df1.head()
```

Out[28]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Exercis
0	-1.43	1	1	0.41	0.83	0	1	1.38	
1	-0.48	0	2	1.49	-0.17	0	1	0.75	
2	-1.75	1	1	-0.13	0.77	0	2	-1.53	
3	-0.58	0	0	0.30	0.14	0	1	-1.13	
4	0.05	1	2	0.95	-0.03	0	1	-0.58	

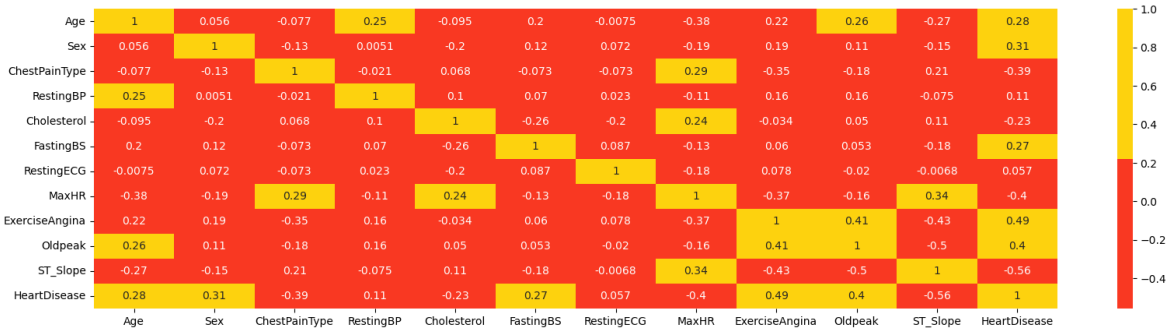
- Machine learning model does not understand the units of the values of the features. It treats the input just as a simple number but does not understand the true meaning of that value. Thus, it becomes necessary to scale the data.

Eg : Age = Years; FastingBS = mg / dl

- We have 2 options for data scaling : 1) **Normalization** 2) **Standardization**. As most of the algorithms assume the data to be normally (Gaussian) distributed, **Normalization** is done for features whose data does not display normal distribution and **standardization** is carried out for features that are normally distributed where their values are huge or very small as compared to other features.
- **Normalization** : **Oldpeak** feature is normalized as it had displayed a right skewed data distribution.
- **Standardizarion** : **Age**, **RestingBP**, **Cholesterol** and **MaxHR** features are scaled down because these features are normally distributed.

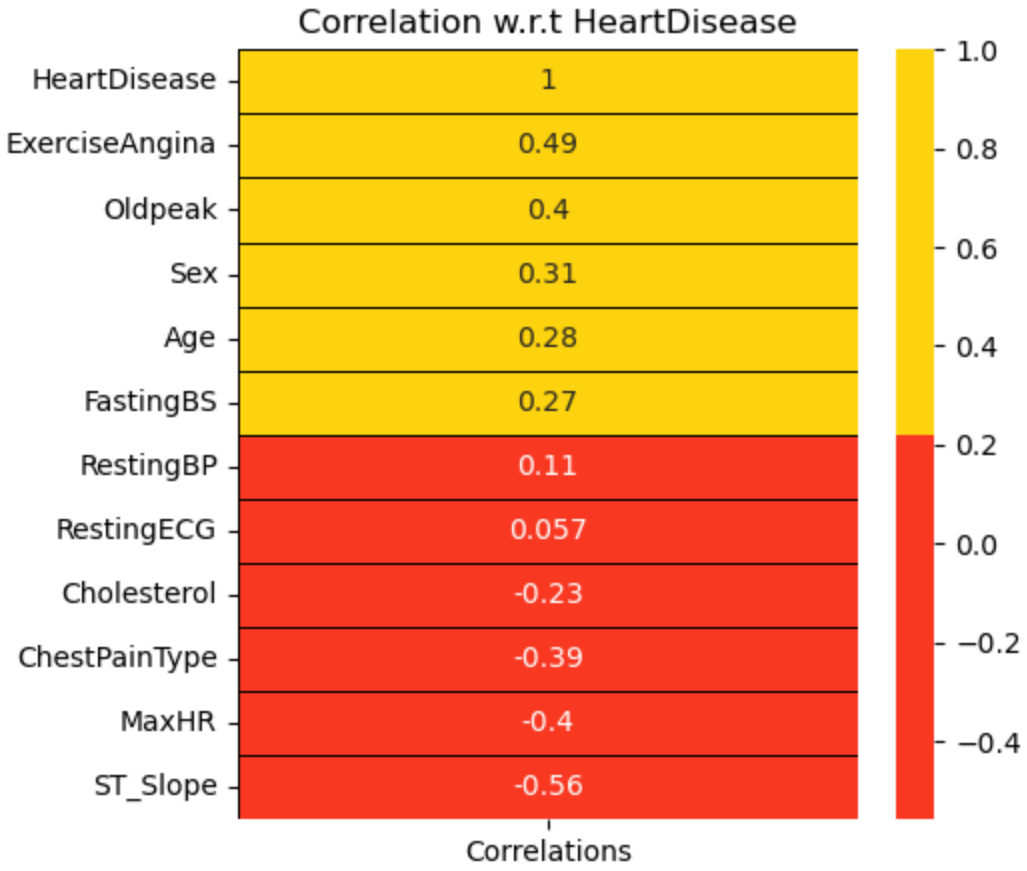
Correlation Matrix :

In [29]:
```
plt.figure(figsize = (20,5))
sns.heatmap(df1.corr(),cmap = colors,annot = True);
```



- It is a huge matrix with too many features. We will check the correlation only with respect to **HeartDisease**.

In [30]:
```
corr = df1.corrwith(df1['HeartDisease']).sort_values(ascending = False).t
corr.columns = ['Correlations']
plt.subplots(figsize = (5,5))
sns.heatmap(corr,annot = True,cmap = colors,linewidths = 0.4,linecolor =
plt.title('Correlation w.r.t HeartDisease');
```



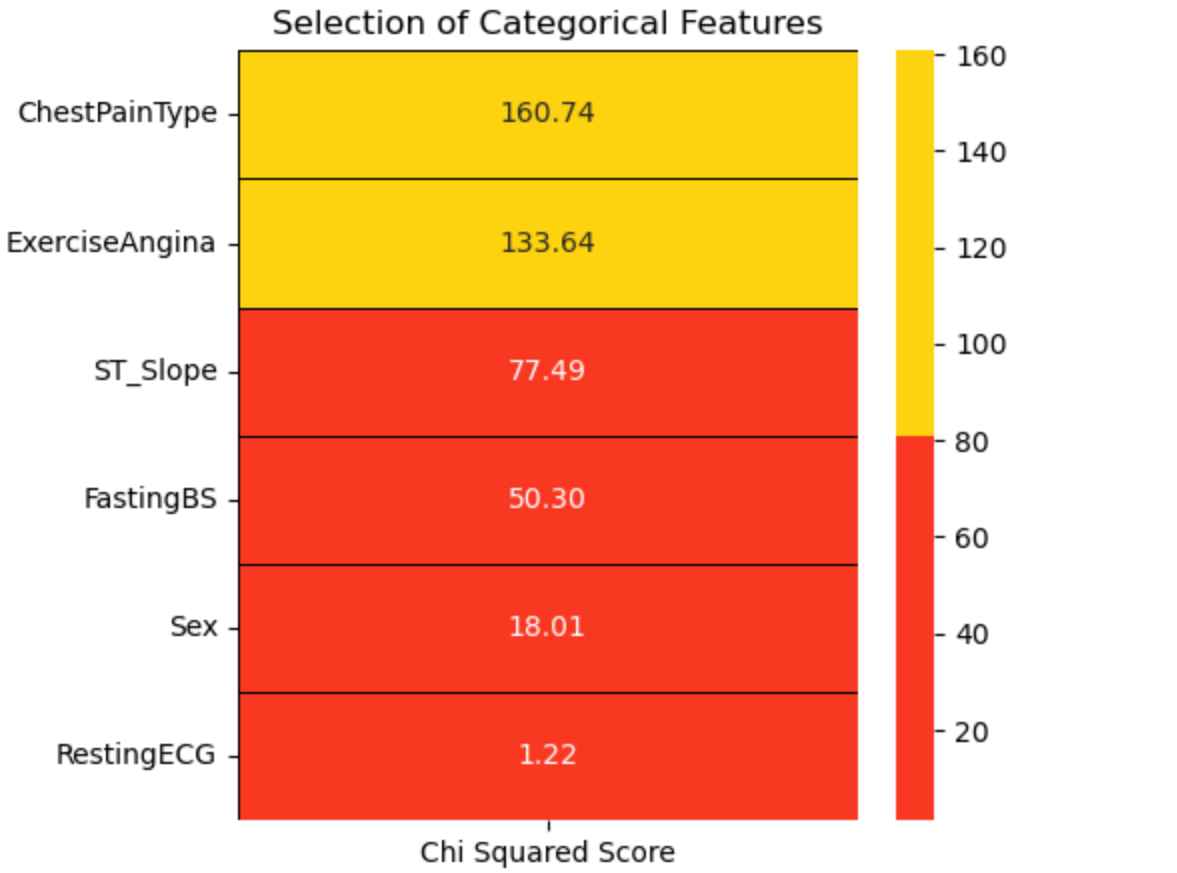
- Except for **RestingBP** and **RestingECG**, everyone displays a positive or negative relationship with **HeartDisease**.

Feature Selection for Categorical Features :

Chi Squared Test :

In [31]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

In [32]:
```
features = df1.loc[:,categorical_features[:-1]]
target = df1.loc[:,categorical_features[-1]]

best_features = SelectKBest(score_func = chi2,k = 'all')
fit = best_features.fit(features,target)

featureScores = pd.DataFrame(data = fit.scores_,index = list(features.col

plt.subplots(figsize = (5,5))
sns.heatmap(featureScores.sort_values(ascending = False,by = 'Chi Squared
plt.title('Selection of Categorical Features');
```

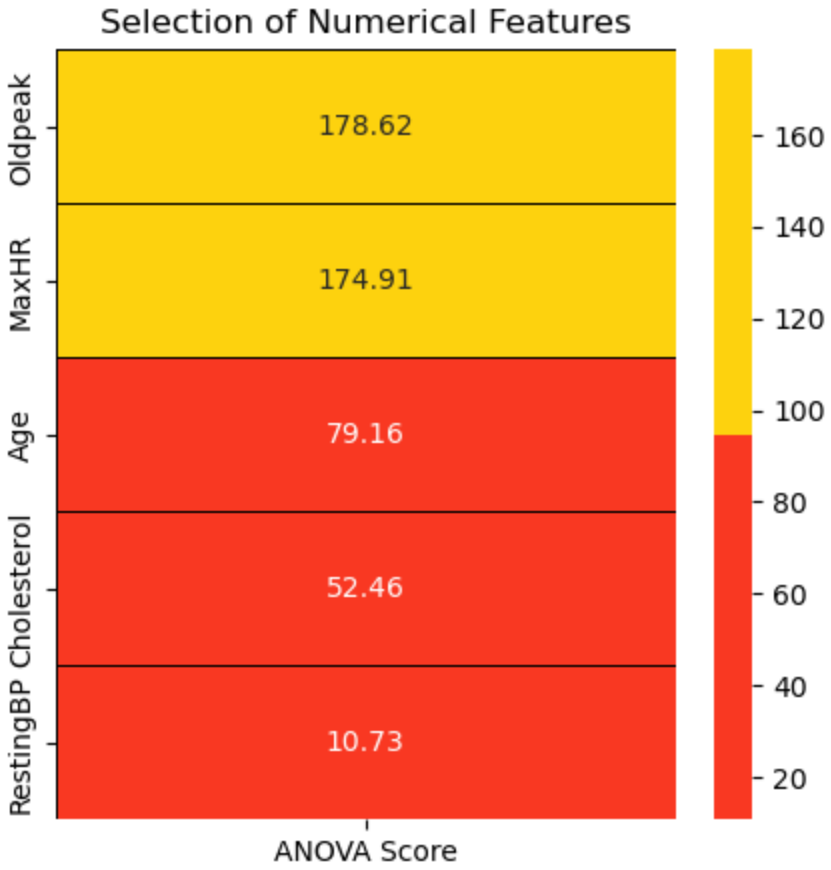


- Except **RestingECG**, all the remaining categorical features are pretty important for predicting heart diseases.

Feature Selection for Numerical Features :

ANOVA Test :

In [33]:
```
from sklearn.feature_selection import f_classif

features = df1.loc[:,numerical_features]
target = df1.loc[:,categorical_features[-1]]

best_features = SelectKBest(score_func = f_classif,k = 'all')
fit = best_features.fit(features,target)

featureScores = pd.DataFrame(data = fit.scores_,index = list(features.col

plt.subplots(figsize = (5,5))
sns.heatmap(featureScores.sort_values(ascending = False,by = 'ANOVA Score
plt.title('Selection of Numerical Features');
```



- We will leave out **RestingBP** from the modeling part and take the remaining features.

Modeling

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

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import precision_recall_curve

import matplotlib.pyplot as plt
```

In [37]:
```
features = df1[df1.columns.drop(['HeartDisease','RestingBP','RestingECG']
target = df1['HeartDisease'].values
x_train, x_test, y_train, y_test = train_test_split(features, target, tes
```

- Selecting the features from the above conducted tests and splitting the data into **80 - 20 train - test** groups.

In [38]:
```
def model(classifier):

    classifier.fit(x_train,y_train)
    prediction = classifier.predict(x_test)
    cv = RepeatedStratifiedKFold(n_splits = 10,n_repeats = 3,random_state
    print("Accuracy : ",'{0:.2%}'.format(accuracy_score(y_test,prediction
    print("Cross Validation Score : ",'{0:.2%}'.format(cross_val_score(cl
    print("ROC_AUC Score : ",'{0:.2%}'.format(roc_auc_score(y_test,predic
    plot_roc_curve(classifier, x_test,y_test)
    plt.title('ROC_AUC_Plot')
    plt.show()

def model_evaluation(classifier):

    # Confusion Matrix
    cm = confusion_matrix(y_test,classifier.predict(x_test))
    names = ['True Neg','False Pos','False Neg','True Pos']
    counts = [value for value in cm.flatten()]
    percentages = ['{0:.2%}'.format(value) for value in cm.flatten()/np.s
    labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(names,counts,perc
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cm,annot = labels,cmap = colors,fmt ='')

    # Classification Report
    print(classification_report(y_test,classifier.predict(x_test)))
```

1] Logistic Regression :

In [39]:
```
from sklearn.linear_model import LogisticRegression
```

In [ ]:
```
from sklearn.metrics import plot_roc_curve
```

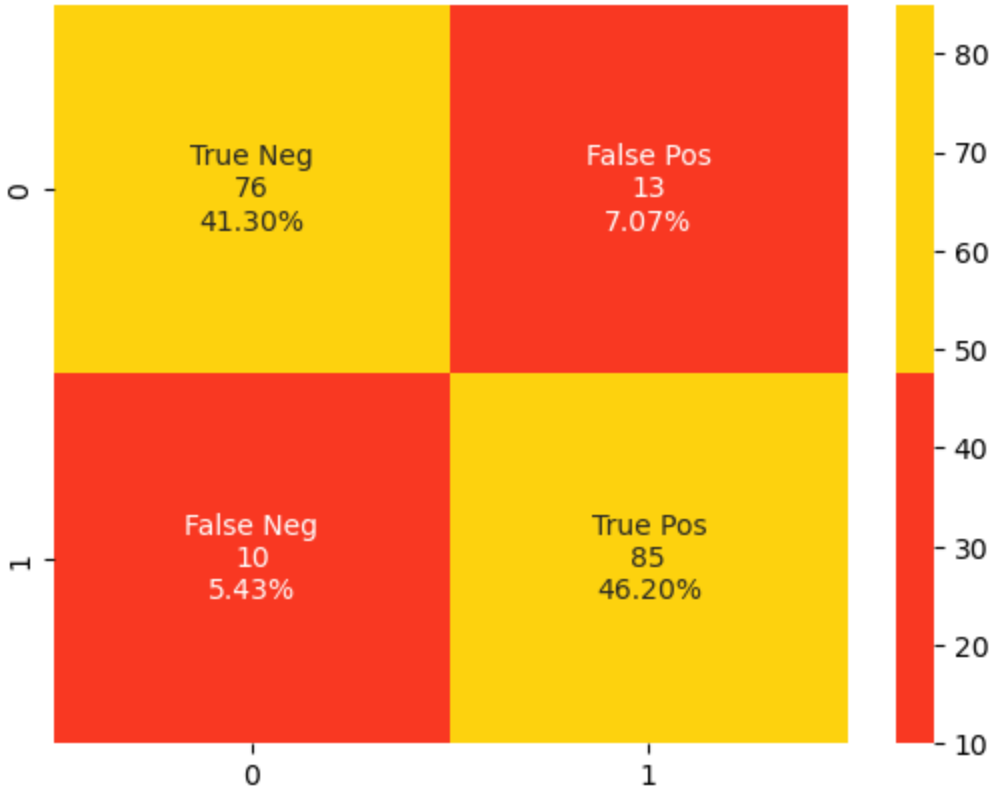
In [40]:
```
classifier_lr = LogisticRegression(random_state = 0,C=10,penalty= 'l2')
```

In [58]:
```
1 model(classifier_lr)
```

```
Accuracy :  87.50%
Cross Validation Score :  91.12%
ROC_AUC Score :  87.43%

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[58], line 1
----> 1 model(classifier_lr)

Cell In[38], line 9, in model(classifier)
      7 print("Cross Validation Score : ",'{0:.2%}'.format(cross_val_score(c
lassifier,x_train,y_train,cv = cv,scoring = 'roc_auc').mean()))
      8 print("ROC_AUC Score : ",'{0:.2%}'.format(roc_auc_score(y_test,predi
ction)))
----> 9 plot_roc_curve(classifier, x_test,y_test)
     10 plt.title('ROC_AUC_Plot')
     11 plt.show()

NameError: name 'plot_roc_curve' is not defined
```

In [59]:
```
1 model_evaluation(classifier_lr)
```

	precision	recall	f1-score	support
0	0.88	0.85	0.87	89
1	0.87	0.89	0.88	95
accuracy			0.88	184
macro avg	0.88	0.87	0.87	184
weighted avg	0.88	0.88	0.87	184



2] Support Vector Classifier :

In [60]:
```
1 from sklearn.svm import SVC
```

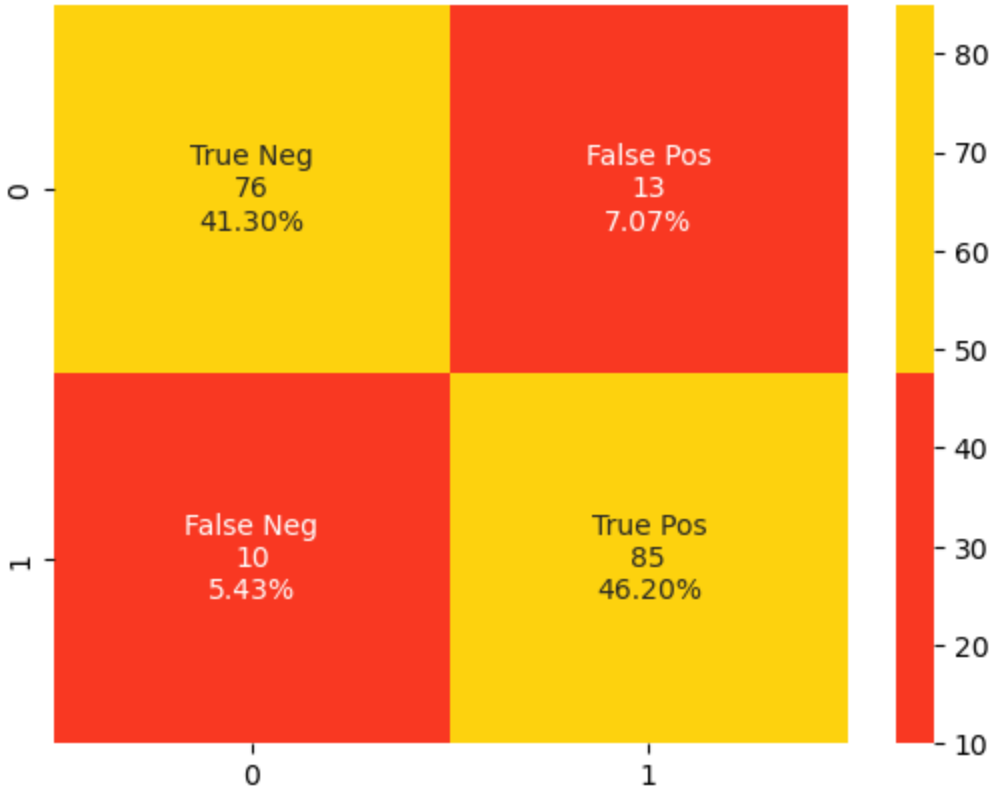
In [61]:
```
1 classifier_svc = SVC(kernel = 'linear',C = 0.1)
```

In [62]:
```
1 model(classifier_svc)
```

```
Accuracy :  87.50%
Cross Validation Score :  90.53%
ROC_AUC Score :  87.43%

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[62], line 1
----> 1 model(classifier_svc)

Cell In[38], line 9, in model(classifier)
      7 print("Cross Validation Score : ",'{0:.2%}'.format(cross_val_score(c
lassifier,x_train,y_train,cv = cv,scoring = 'roc_auc').mean()))
      8 print("ROC_AUC Score : ",'{0:.2%}'.format(roc_auc_score(y_test,predi
ction)))
----> 9 plot_roc_curve(classifier, x_test,y_test)
     10 plt.title('ROC_AUC_Plot')
     11 plt.show()

NameError: name 'plot_roc_curve' is not defined
```

In [63]:
```
1 model_evaluation(classifier_svc)
```

```
              precision    recall  f1-score   support

           0       0.88      0.85      0.87        89
           1       0.87      0.89      0.88        95

    accuracy                           0.88       184
   macro avg       0.88      0.87      0.87       184
weighted avg       0.88      0.88      0.87       184
```



3] Decision Tree Classifier :

In [47]:
```
1 from sklearn.tree import DecisionTreeClassifier
```

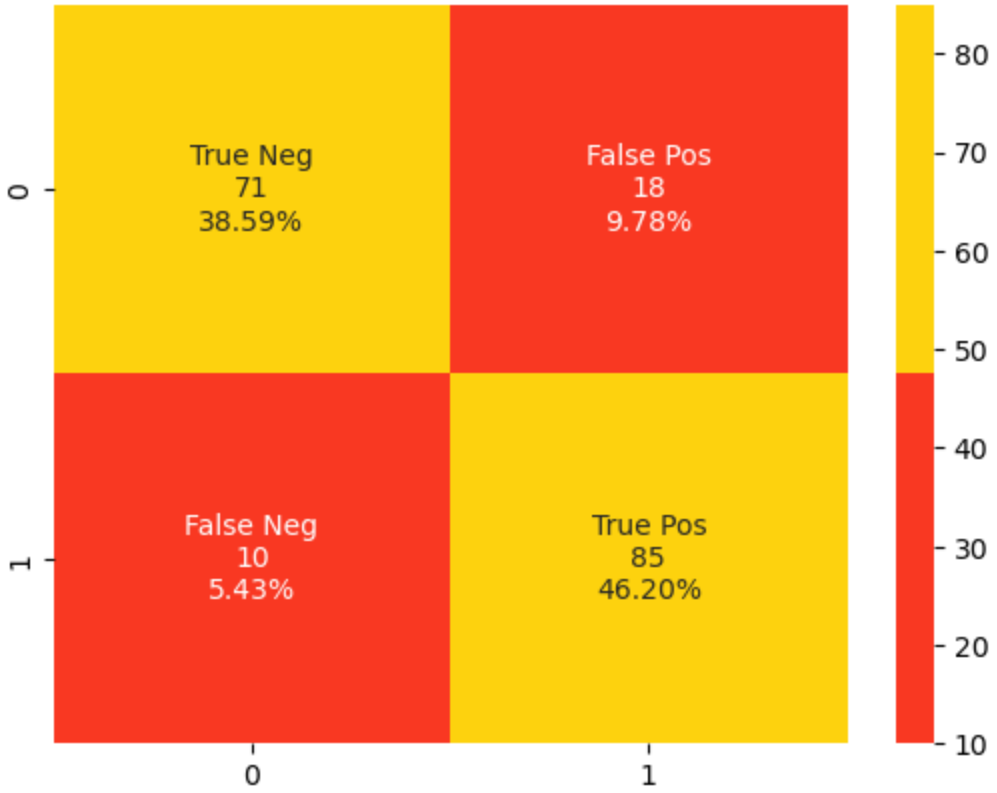
In [48]:
```
1 classifier_dt = DecisionTreeClassifier(random_state = 1000,max_depth = 4,
```

In [49]:
```
1 model(classifier_dt)
```

```
Accuracy :  84.78%
Cross Validation Score :  89.09%
ROC_AUC Score :  84.62%

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[49], line 1
----> 1 model(classifier_dt)

Cell In[38], line 9, in model(classifier)
      7 print("Cross Validation Score : ",'{0:.2%}'.format(cross_val_score(c
lassifier,x_train,y_train,cv = cv,scoring = 'roc_auc').mean()))
      8 print("ROC_AUC Score : ",'{0:.2%}'.format(roc_auc_score(y_test,predi
ction)))
----> 9 plot_roc_curve(classifier, x_test,y_test)
     10 plt.title('ROC_AUC_Plot')
     11 plt.show()

NameError: name 'plot_roc_curve' is not defined
```

In [50]:
```
1 model_evaluation(classifier_dt)
```

```
              precision    recall  f1-score   support

           0       0.88      0.80      0.84        89
           1       0.83      0.89      0.86        95

    accuracy                           0.85       184
   macro avg       0.85      0.85      0.85       184
weighted avg       0.85      0.85      0.85       184
```



4] Random Forest Classifier :

In [51]:
```
1 from sklearn.ensemble import RandomForestClassifier
```

In [52]:
```
1 classifier_rf = RandomForestClassifier(max_depth = 4,random_state = 0)
```

In [53]:
```
1 model(classifier_rf)
```

```
Accuracy :  84.24%
Cross Validation Score :  92.91%
ROC_AUC Score :  84.06%

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[53], line 1
----> 1 model(classifier_rf)

Cell In[38], line 9, in model(classifier)
      7 print("Cross Validation Score : ",'{0:.2%}'.format(cross_val_score(c
lassifier,x_train,y_train,cv = cv,scoring = 'roc_auc').mean()))
      8 print("ROC_AUC Score : ",'{0:.2%}'.format(roc_auc_score(y_test,predi
ction)))
----> 9 plot_roc_curve(classifier, x_test,y_test)
     10 plt.title('ROC_AUC_Plot')
     11 plt.show()

NameError: name 'plot_roc_curve' is not defined
```

In [ ]:
```
1 model_evaluation(classifier_rf)
```

5] K-nearest Neighbors Classifier :

In [54]:
```
1 from sklearn.neighbors import KNeighborsClassifier
```

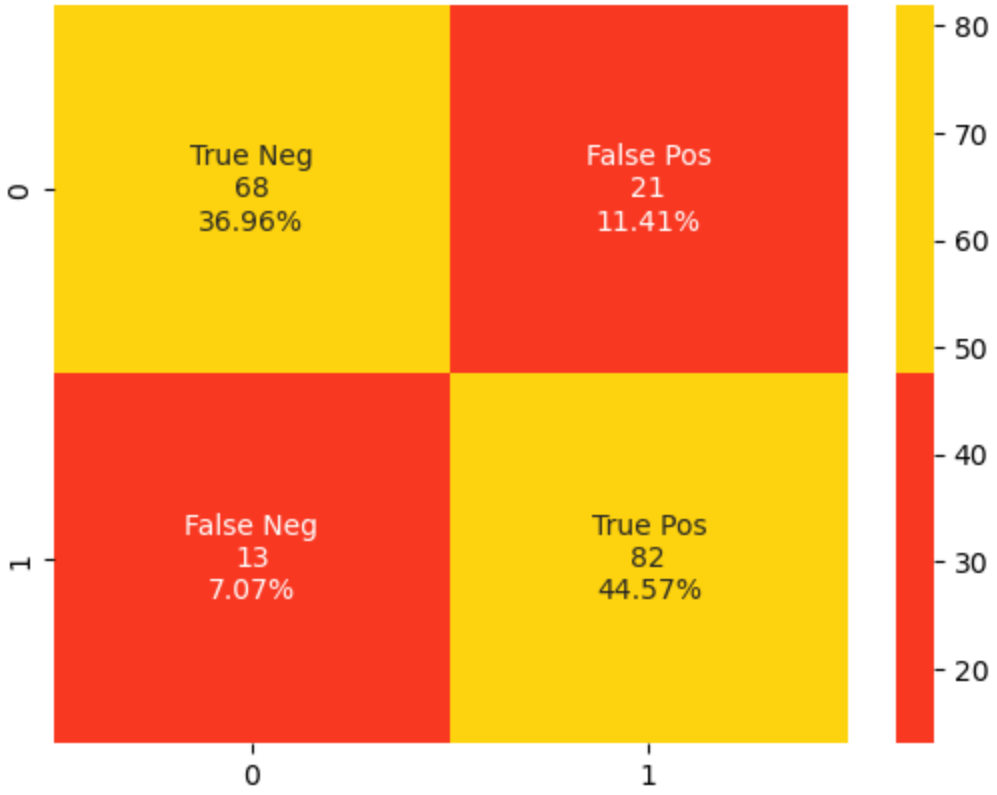
In [55]:
```
1 classifier_knn = KNeighborsClassifier(leaf_size = 1, n_neighbors = 3,p =
```

In [56]:
```
1 model(classifier_knn)
```

```
Accuracy :  81.52%
Cross Validation Score :  89.34%
ROC_AUC Score :  81.36%

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[56], line 1
----> 1 model(classifier_knn)

Cell In[38], line 9, in model(classifier)
      7 print("Cross Validation Score : ",'{0:.2%}'.format(cross_val_score(c
lassifier,x_train,y_train,cv = cv,scoring = 'roc_auc').mean()))
      8 print("ROC_AUC Score : ",'{0:.2%}'.format(roc_auc_score(y_test,predi
ction)))
----> 9 plot_roc_curve(classifier, x_test,y_test)
     10 plt.title('ROC_AUC_Plot')
     11 plt.show()

NameError: name 'plot_roc_curve' is not defined
```

In [57]:
```
1 model_evaluation(classifier_knn)
```

```
              precision    recall  f1-score   support

           0       0.84      0.76      0.80        89
           1       0.80      0.86      0.83        95

    accuracy                           0.82       184
   macro avg       0.82      0.81      0.81       184
weighted avg       0.82      0.82      0.81       184
```



Alogrithm Results Table :

Sr. No.	ML Algorithm	Accuracy	Cross Validation Score	ROC AUC Score
1	Logistic Regression	87.50%	91.12%	87.43%
2	Support Vector Classifier	87.50%	90.53%	87.43%
3	Decision Tree Classifier	84.78%	89.09%	84.62%
4	Random Forest Classifier	84.24%	92.91%	84.06%
5	K-Nearest Neighbors Classfier	81.52%	89.34%	81.36%

In [ ]:
```
1
```