NAME: ANJALI RAMAN

STUDENT ID: 11521434

In [1]:

```
# Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, f1_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
# Load the dataset
df = pd.read_csv("Heart_Disease_Prediction Dataset.csv")
df.head()
#https://www.kaggle.com/datasets/thedevastator/predicting-heart-disease-risk-using-clinical-var.
```

Out[2]:

	index	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Hea Diseas
0	0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presenc
1	1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absenc
2	2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presenc
3	3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absenc
4	4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absenc

In [3]:

```
df.columns
```

Out[3]:

```
Index(['index', 'Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol',
       'FBS over 120', 'EKG results', 'Max HR', 'Exercise angina',
       'ST depression', 'Slope of ST', 'Number of vessels fluro', 'Thallium',
       'Heart Disease'],
      dtype='object')
```

In [4]:

```
df.shape
```

Out[4]:

```
(270, 15)
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   index                    270 non-null    int64  
 1   Age                      270 non-null    int64  
 2   Sex                      270 non-null    int64  
 3   Chest pain type          270 non-null    int64  
 4   BP                       270 non-null    int64  
 5   Cholesterol              270 non-null    int64  
 6   FBS over 120             270 non-null    int64  
 7   EKG results              270 non-null    int64  
 8   Max HR                   270 non-null    int64  
 9   Exercise angina          270 non-null    int64  
 10  ST depression            270 non-null    float64
 11  Slope of ST              270 non-null    int64  
 12  Number of vessels fluro  270 non-null    int64  
 13  Thallium                 270 non-null    int64  
 14  Heart Disease            270 non-null    object 
dtypes: float64(1), int64(13), object(1)
memory usage: 31.8+ KB
```

In [6]:

```
df.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
index	270.0	134.500000	78.086491	0.0	67.25	134.5	201.75	269.0
Age	270.0	54.433333	9.109067	29.0	48.00	55.0	61.00	77.0
Sex	270.0	0.677778	0.468195	0.0	0.00	1.0	1.00	1.0
Chest pain type	270.0	3.174074	0.950090	1.0	3.00	3.0	4.00	4.0
BP	270.0	131.344444	17.861608	94.0	120.00	130.0	140.00	200.0
Cholesterol	270.0	249.659259	51.686237	126.0	213.00	245.0	280.00	564.0
FBS over 120	270.0	0.148148	0.355906	0.0	0.00	0.0	0.00	1.0
EKG results	270.0	1.022222	0.997891	0.0	0.00	2.0	2.00	2.0
Max HR	270.0	149.677778	23.165717	71.0	133.00	153.5	166.00	202.0
Exercise angina	270.0	0.329630	0.470952	0.0	0.00	0.0	1.00	1.0
ST depression	270.0	1.050000	1.145210	0.0	0.00	0.8	1.60	6.2
Slope of ST	270.0	1.585185	0.614390	1.0	1.00	2.0	2.00	3.0
Number of vessels fluro	270.0	0.670370	0.943896	0.0	0.00	0.0	1.00	3.0
Thallium	270.0	4.696296	1.940659	3.0	3.00	3.0	7.00	7.0

In [7]:

```
df.describe(include='object').T
```

Out[7]:

	count	unique	top	freq
Heart Disease	270	2	Absence	150

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
index                     0
Age                       0
Sex                       0
Chest pain type           0
BP                        0
Cholesterol               0
FBS over 120              0
EKG results               0
Max HR                    0
Exercise angina           0
ST depression             0
Slope of ST               0
Number of vessels fluro   0
Thallium                  0
Heart Disease             0
dtype: int64
```

In [9]:

```
df['Heart Disease'].value_counts()
```

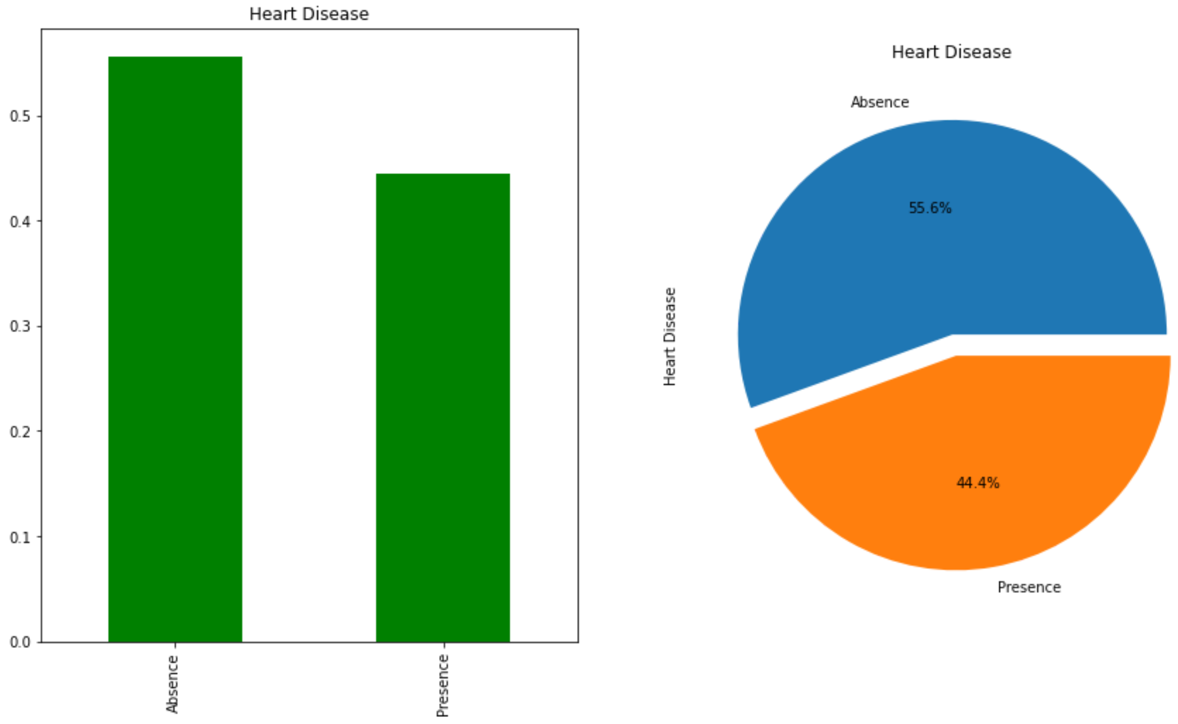
Out[9]:

```
Absence     150
Presence    120
Name: Heart Disease, dtype: int64
```

The output of the dataset describes the numerical values for Presence and Absence of Heart Disease resulting in 150 for Absence of Heart Disease and 120 for Presence.

In [10]:

```
plt.figure(figsize=(15, 8))
plt.subplot(1,2,1)
df['Heart Disease'].value_counts(normalize = True).plot(kind = 'bar', color = 'green')
plt.title('Heart Disease')
plt.subplot(1,2,2)
plt.title('Heart Disease')
df['Heart Disease'].value_counts().plot(kind = 'pie', autopct='%1.1f%%', explode = [0, 0.1])
plt.show()
```



The output is the graphical representation showing Bar Graph and a Pie chart describing the information of Presence and Absence of Heart Disease. The Bar Graph shows the percentage of Heart Disease on Y-axis and attributes of Heart Disease on X-axis. The Pie chart shows the same information with Absence with 55.6% while Presence of Heart Disease with 44.4%.

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   index                    270 non-null    int64  
 1   Age                      270 non-null    int64  
 2   Sex                      270 non-null    int64  
 3   Chest pain type          270 non-null    int64  
 4   BP                       270 non-null    int64  
 5   Cholesterol              270 non-null    int64  
 6   FBS over 120             270 non-null    int64  
 7   EKG results              270 non-null    int64  
 8   Max HR                   270 non-null    int64  
 9   Exercise angina          270 non-null    int64  
 10  ST depression            270 non-null    float64
 11  Slope of ST              270 non-null    int64  
 12  Number of vessels fluro  270 non-null    int64  
 13  Thallium                 270 non-null    int64  
 14  Heart Disease            270 non-null    object 
dtypes: float64(1), int64(13), object(1)
memory usage: 31.8+ KB
```

The Output shows dataset with total of 270 data entires divided into 14 columns. The output also shows that there are zero null values in the dataset. With Heart Disease as Target variable.

In [12]:

```
df.head()
```

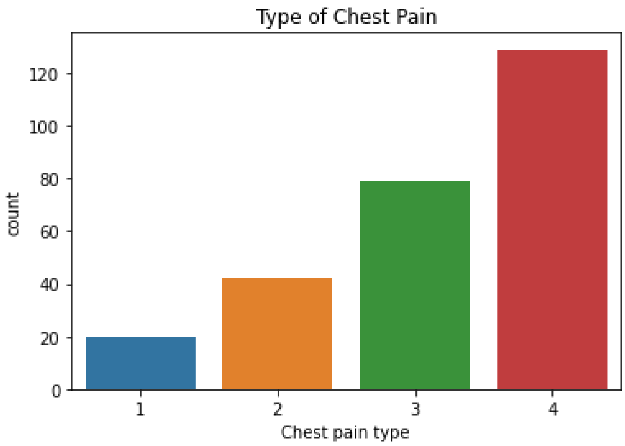
Out[12]:

	index	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Hea Diseas
0	0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presenc
1	1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absenc
2	2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presenc
3	3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absenc
4	4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absenc

Exploratory Data Analysis(EDA)

In [13]:

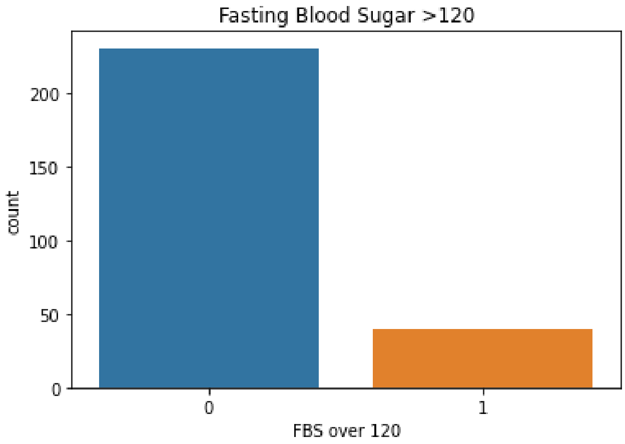
```
sns.countplot(df["Chest pain type"])
plt.title('Type of Chest Pain')
plt.show()
```



The output Bar Graph represents the variety of chest pain and count of it. X-axis represents the Type of chest pain and Y-axis represents Count of it.

In [14]:

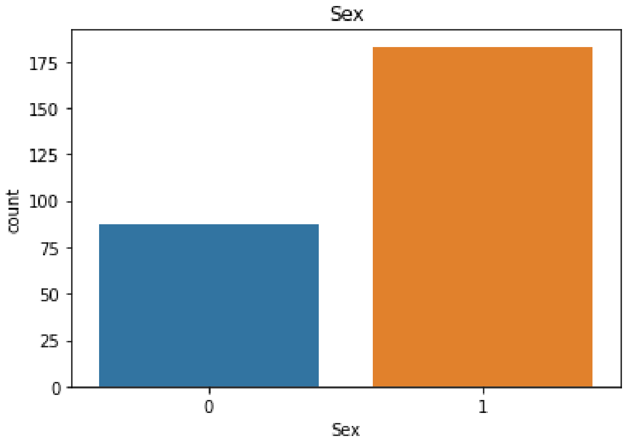
```
sns.countplot(df["FBS over 120"])
plt.title('Fasting Blood Sugar >120')
plt.show()
```



The Output bar graph here represents count of people with FBS>120 where 1 represents FBS>120 and 0 represents FBS<120. Here, X-axis is represented by the value of FBS and Y-axis is represented by Count

In [15]:

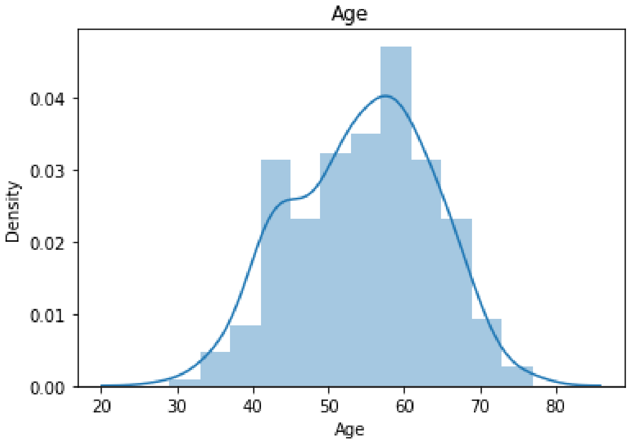
```
sns.countplot(df["Sex"])
plt.title('Sex')
plt.show()
```



Here the output shows, the relation for Sex and Count of Heart disease having 1 as 'Male' and 0 as 'Female'. We can observe that male tend to have more chance to get Heart Disease when compared to female.

In [16]:

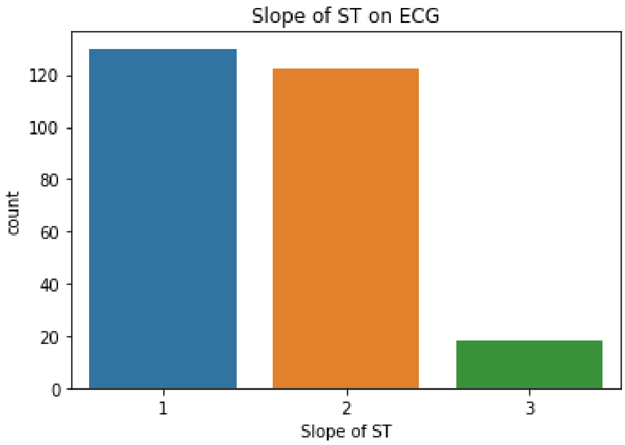
```
sns.distplot(df['Age'])
plt.title('Age')
plt.show()
```



The output is a Histogram represetation that explains the distribution of heart disease as per Age and Density of Disease having 'Age' on X-axis and 'Density' on Y-axis. By thus representation we can say that people between age groups 40-65 have high risk of Heart Disease.

In [17]:

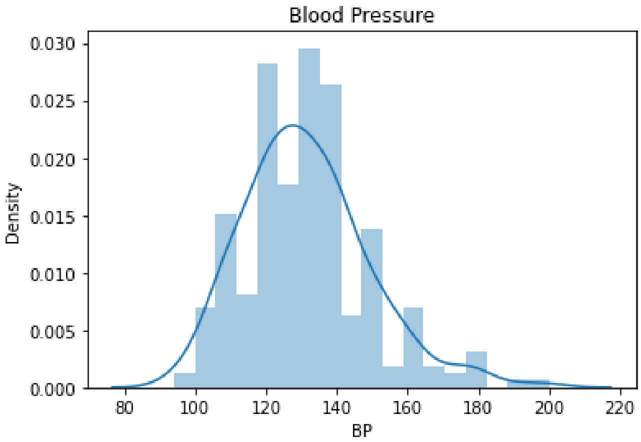
```
sns.countplot(df["Slope of ST"])
plt.title('Slope of ST on ECG')
plt.show()
```



The out is the Bar Graph representation of distribution of data between slope of ST and Count of Heart Disease. On x-axis, 1,2 and 3 represent Downsloping, Flat, and Upsloping respectively.

In [18]:

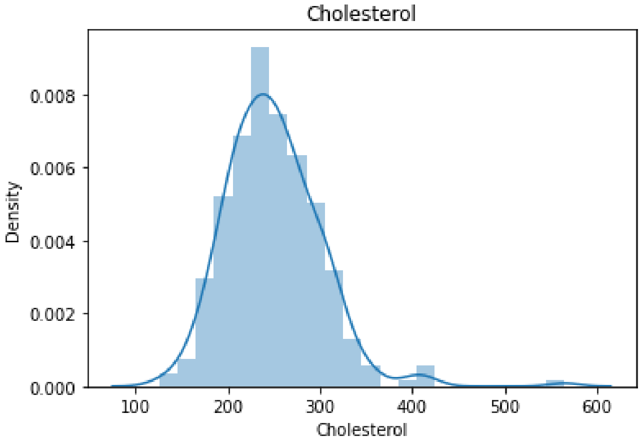
```
sns.distplot(df['BP'])
plt.title('Blood Pressure')
plt.show()
```



This plot represents the distribution of Blood pressure. In this case, we can say that the peak is between 120-130mm Hg with density between 0.020-0.025. We can also say that patients with High blood pressure are more from the dataset.

In [19]:

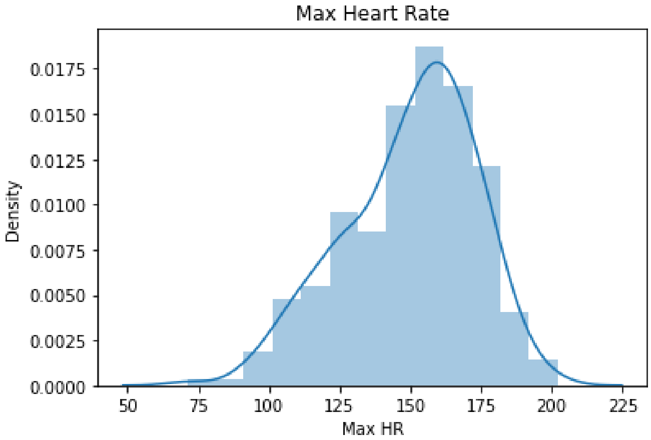
```
sns.distplot(df['Cholesterol'])
plt.title('Cholesterol')
plt.show()
```



The output histogram represents values of cholesterol with peak between 200-250mg/dl. With peak distribution we can say that according to the dataset, there are more patients with high cholesterol levels.

In [20]:

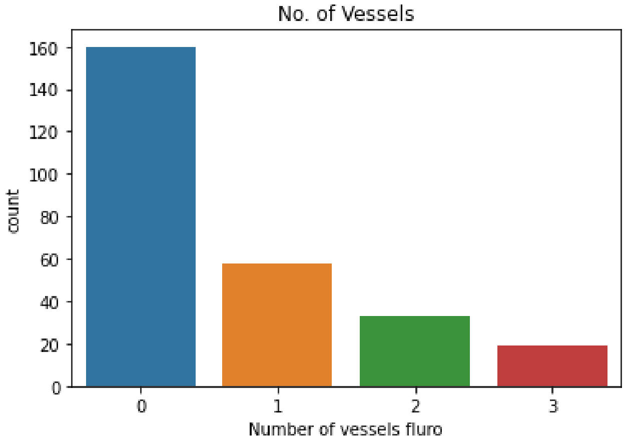
```
sns.distplot(df['Max HR'])
plt.title('Max Heart Rate')
plt.show()
```



The Histogram represents the values of patients with Maximum Heart Rate. From the graph we can say that the distribution with peak between 150-170 Bpm is high. This plot also represents that patients with Max HR are less.

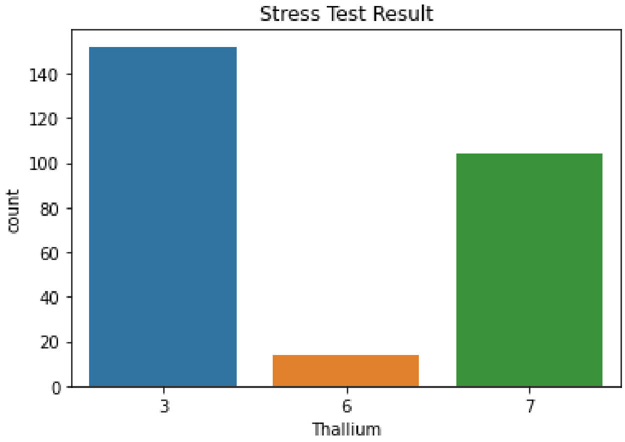
In [21]:

```
sns.countplot(df["Number of vessels fluro"])
plt.title('No. of Vessels')
plt.show()
```



In [22]:

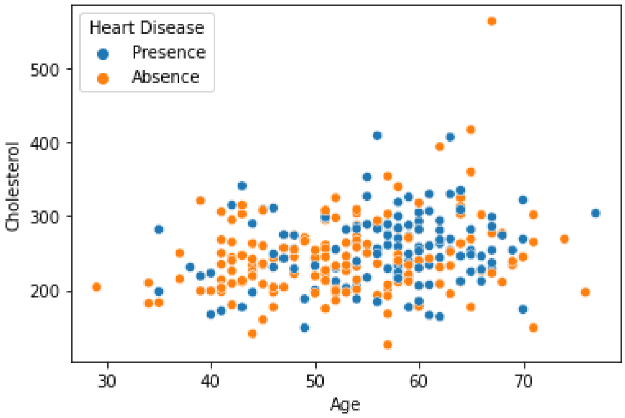
```
sns.countplot(df["Thallium"])
plt.title('Stress Test Result')
plt.show()
```



Scatter Plot Representations

In [23]:

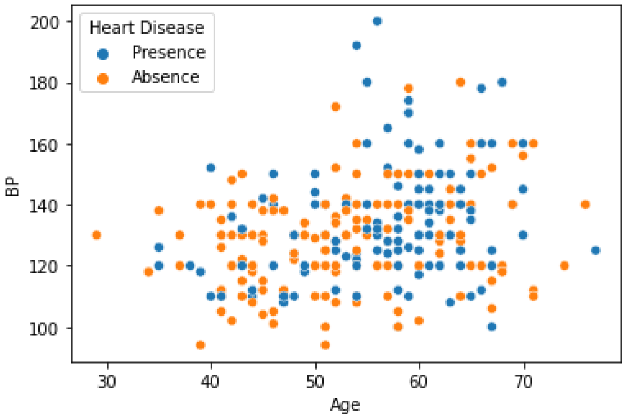
```
sns.scatterplot(df['Age'],df['Cholesterol'],hue = df['Heart Disease'])
plt.show()
```



The out Scatter Plot represents the relation between Age and Cholesterol and Heart Disease. By this plot we can say that patients in

In [24]:

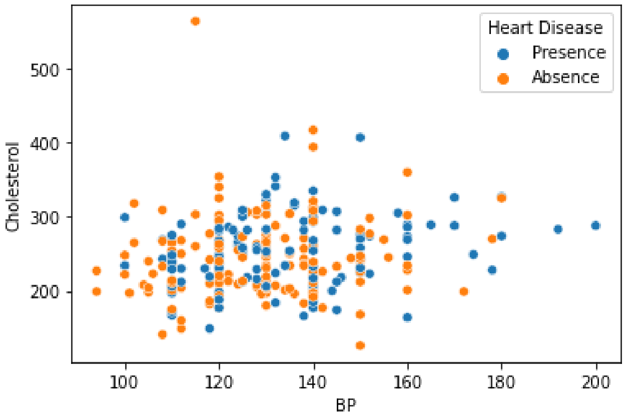
```python
sns.scatterplot(df['Age'],df['BP'],hue = df['Heart Disease'])
plt.show()
```



The out Scatter Plot represents the relation between Age and BP and Heart Disease. Using this plot we can say that patiets with Heart disease present tend to have high BP levels. But, by this scatter plot it is difficult to predict Heart Disease using just the levels of BP.

In [25]:

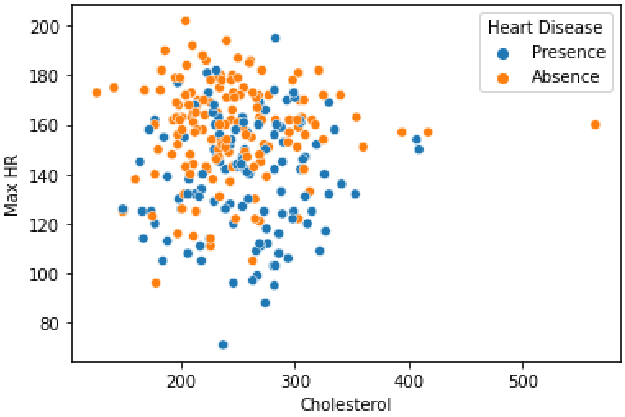
```python
sns.scatterplot(df['BP'],df['Cholesterol'],hue = df['Heart Disease'])
plt.show()
```



The above scatter plot represents the relation between BP, Cholesterol adn Heart Disease. From the plot, we can say that BP and Cholesterol are highly correlated and Heart Disease can be present in patients with High BP and Cholesterol

In [26]:

```python
sns.scatterplot(df['Cholesterol'],df['Max HR'],hue = df['Heart Disease'])
plt.show()
```

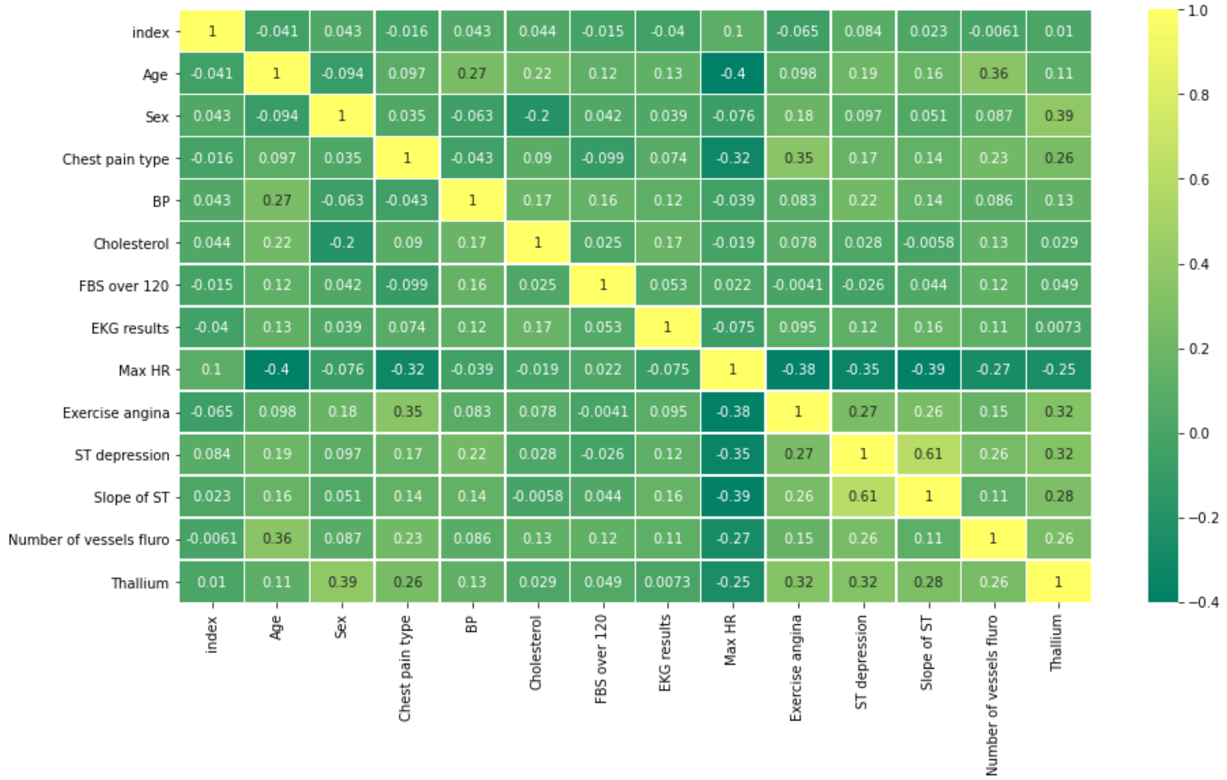


The Output Scatter plot represents relation between Cholesterol, Max Heart RAte and Heart Disease. This plot shows that the patients with heart disease can have high cholesterol when compared to patients without heart disease.

HEAT MAP

In [27]:

```
plt.figure(figsize=(15,8))
sns.heatmap(df.corr(), annot = True, linewidth = 0.5, cmap='summer')
plt.show()
```



ENCODING DATA

In [28]:

```
encoder = LabelEncoder()
df["Heart Disease"] = encoder.fit_transform(df["Heart Disease"])
```

In [29]:

```
df.head()
```

Out[29]:

	index	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Hear Diseas
0	0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	'
1	1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	(
2	2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	'
3	3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	(
4	4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	(

In [30]:

```
scaler = StandardScaler()
df[["BP", "Cholesterol", "Max HR"]] = scaler.fit_transform(df[["BP", "Cholesterol", "Max HR"]])
```

In [31]:

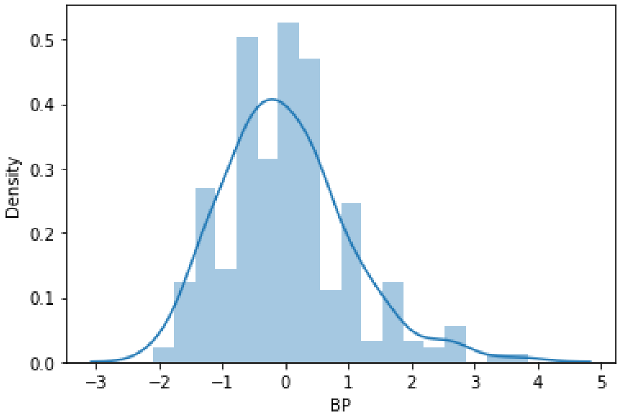
```
df.head()
```

Out[31]:

	index	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thalliu
0	0	70	1	4	-0.075410	1.402212	0	2	-1.759208	0	2.4	2	3	
1	1	67	0	3	-0.916759	6.093004	0	2	0.446409	0	1.6	2	0	
2	2	57	1	2	-0.411950	0.219823	0	0	-0.375291	0	0.3	1	0	
3	3	64	1	4	-0.187590	0.258589	0	0	-1.932198	1	0.2	2	1	
4	4	74	0	2	-0.636310	0.374890	0	2	-1.240239	1	0.2	1	1	

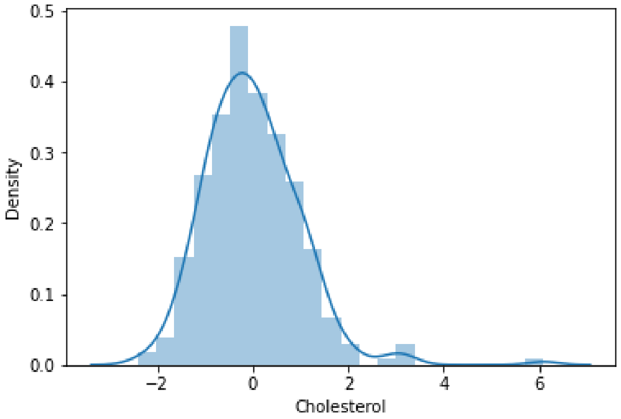
In [32]:

```
sns.distplot(df['BP'])
plt.show()
```



In [33]:

```
sns.distplot(df['Cholesterol'])
plt.show()
```



In [34]:

```
df=df.drop('Max HR', axis=1)
```

In [35]:

```
df.head()
```

Out[35]:

	index	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Heart Disease
0	0	70	1	4	-0.075410	1.402212	0	2	0	2.4	2	3	3	1
1	1	67	0	3	-0.916759	6.093004	0	2	0	1.6	2	0	7	0
2	2	57	1	2	-0.411950	0.219823	0	0	0	0.3	1	0	7	1
3	3	64	1	4	-0.187590	0.258589	0	0	1	0.2	2	1	7	0
4	4	74	0	2	-0.636310	0.374890	0	2	1	0.2	1	1	3	0

In [36]:

```
y=df['Heart Disease']
x=df.drop('Heart Disease', axis=1)
```

In [37]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=20)
print('The shape of the x_train:',x_train.shape)
print('The shape of the x_test:',x_test.shape)
print('The shape of the y_train:',y_train.shape)
print('The shape of the y_test:',y_test.shape)
```

```
The shape of the x_train: (189, 13)
The shape of the x_test: (81, 13)
The shape of the y_train: (189,)
The shape of the y_test: (81,)
```

In [38]:

```
lr = LogisticRegression(solver='saga')
df_lr = lr.fit(x_train, y_train)
df_lr_pred_test = df_lr.predict(x_test)
df_lr_pred_train = df_lr.predict(x_train)
df_lr_prob = df_lr.predict_proba(x_test)[:,1]
```

In [39]:

```
lr_acc_score_test = print('The test accuracy score of Logistic Regression is: ', accuracy_score(y_test,df_l
lr_acc_score_test
```

```
The test accuracy score of Logistic Regression is:  74.07407407407408
```

In [40]:

```
lr_acc_score_train = print('The train accuracy score of Logistic Regression is: ', accuracy_score(y_train,d
lr_acc_score_train
```

```
The train accuracy score of Logistic Regression is:  77.24867724867724
```

In [41]:

```
print('The f1 score of Logistic Regression is: ', f1_score(y_test,df_lr_pred_test)*100)
```

```
The f1 score of Logistic Regression is:  69.56521739130434
```

In [42]:

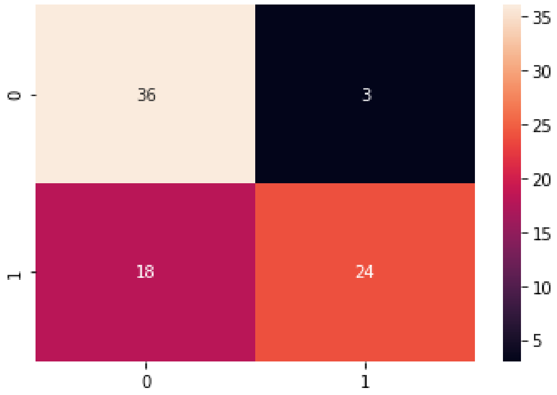
```
print('Classification report : \n',classification_report(y_test,df_lr_pred_test))
print('confusion matrix : \n',confusion_matrix(y_test,df_lr_pred_test))
sns.heatmap(confusion_matrix(y_test,df_lr_pred_test), annot = True)
plt.show()
```

```
Classification report :
               precision    recall  f1-score   support

           0       0.67      0.92      0.77        39
           1       0.89      0.57      0.70        42

    accuracy                           0.74        81
   macro avg       0.78      0.75      0.73        81
weighted avg       0.78      0.74      0.73        81

confusion matrix :
 [[36  3]
 [18 24]]
```



In [43]:

```
dtc = DecisionTreeClassifier()
df_dtc = dtc.fit(x_train, y_train)
df_dtc_pred_test = df_dtc.predict(x_test)
df_dtc_pred_train = df_dtc.predict(x_train)
df_dtc_prob = df_dtc.predict_proba(x_test)[:,1]
```

In [44]:

```
dtc_acc_score_test = print('The test accuracy score of Decision Tree is: ', accuracy_score(y_test,df_dtc_pr
dtc_acc_score_test
```

```
The test accuracy score of Decision Tree is:  76.5432098765432
```

In [45]:

```
dtc_acc_score_train = print('The train accuracy score of Decision Tree is: ', accuracy_score(y_train,df_dtc_
dtc_acc_score_train
```

```
The train accuracy score of Decision Tree is:  100.0
```

In [46]:

```
print('The f1 score of Decision Tree is: ', f1_score(y_test,df_dtc_pred_test)*100)
```

```
The f1 score of Decision Tree is:  78.65168539325842
```

In [47]:

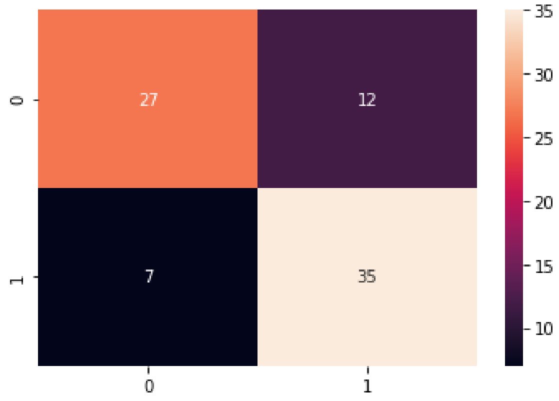
```
print('Classification report : \n',classification_report(y_test,df_dtc_pred_test))
print('confusion matrix : \n',confusion_matrix(y_test,df_dtc_pred_test))
sns.heatmap(confusion_matrix(y_test,df_dtc_pred_test), annot = True)
plt.show()
```

```
Classification report :
               precision    recall  f1-score   support

           0       0.79      0.69      0.74        39
           1       0.74      0.83      0.79        42

    accuracy                           0.77        81
   macro avg       0.77      0.76      0.76        81
weighted avg       0.77      0.77      0.76        81

confusion matrix :
 [[27 12]
 [ 7 35]]
```



In [48]:

```
rfc = RandomForestClassifier()
df_rfc = rfc.fit(x_train, y_train)
df_rfc_pred_test = df_rfc.predict(x_test)
df_rfc_pred_train = df_rfc.predict(x_train)
df_rfc_prob = df_rfc.predict_proba(x_test)[:,1]
```

In [49]:

```
rfc_acc_score_test = print('The test accuracy score of Random Forrest is: ', accuracy_score(y_test,df_rfc_p
rfc_acc_score_test
```

```
The test accuracy score of Random Forrest is:  83.9506172839506
```

In [50]:

```
rfc_acc_score_train = print('The train accuracy score of Random Forrest is: ', accuracy_score(y_train,df_rf
rfc_acc_score_train
```

```
The train accuracy score of Random Forrest is:  100.0
```

In [51]:

```
print('The f1 score of Random Forrest is: ', f1_score(y_test,df_rfc_pred_test)*100)
```

```
The f1 score of Random Forrest is:  83.95061728395062
```

In [52]:

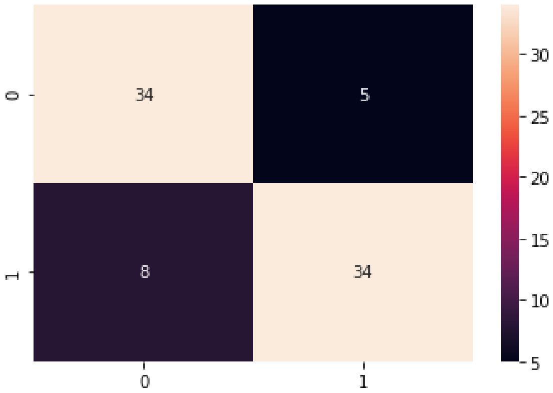
```
print('Classification report : \n',classification_report(y_test,df_rfc_pred_test))
print('confusion matrix : \n',confusion_matrix(y_test,df_rfc_pred_test))
sns.heatmap(confusion_matrix(y_test,df_rfc_pred_test), annot = True)
plt.show()
```

```
Classification report :
               precision    recall  f1-score   support

           0       0.81      0.87      0.84        39
           1       0.87      0.81      0.84        42

    accuracy                           0.84        81
   macro avg       0.84      0.84      0.84        81
weighted avg       0.84      0.84      0.84        81

confusion matrix :
 [[34  5]
 [ 8 34]]
```



In [53]:

```
knn = KNeighborsClassifier()
df_knn = knn.fit(x_train, y_train)
df_knn_pred_test = df_knn.predict(x_test)
df_knn_pred_train = df_knn.predict(x_train)
df_knn_prob = df_knn.predict_proba(x_test)[:,1]
```

In [54]:

```
knn_acc_score_test = print('The test accuracy score of KNN is: ', accuracy_score(y_test,df_knn_pred_test)*1
knn_acc_score_test
```

```
The test accuracy score of KNN is:  53.086419753086425
```

In [55]:

```
knn_acc_score_train = print('The train accuracy score of KNN is: ', accuracy_score(y_train,df_knn_pred_trai
knn_acc_score_train
```

```
The train accuracy score of KNN is:  75.13227513227513
```

In [56]:

```
print('The f1 score of KNN is: ', f1_score(y_test,df_knn_pred_test)*100)
```

```
The f1 score of KNN is:  53.658536585365844
```

In [57]:

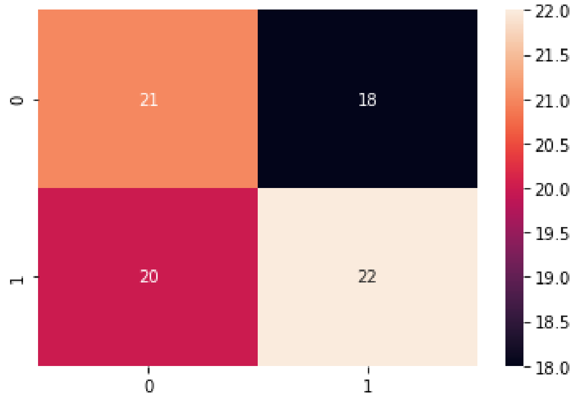
```
print('Classification report : \n',classification_report(y_test,df_knn_pred_test))
print('confusion matrix : \n',confusion_matrix(y_test,df_knn_pred_test))
sns.heatmap(confusion_matrix(y_test,df_knn_pred_test), annot = True)
plt.show()
```

```
Classification report :
               precision    recall  f1-score   support

           0       0.51      0.54      0.53        39
           1       0.55      0.52      0.54        42

    accuracy                           0.53        81
   macro avg       0.53      0.53      0.53        81
weighted avg       0.53      0.53      0.53        81

confusion matrix :
 [[21 18]
 [20 22]]
```



In [ ]: