ASSIGNMENT 4

- Name : Achal Rajesh Mate
- Roll No : 2203541
- Enroll No : MITU20BTCSD001
- Branch : CSE
- Class : TY CSE Is - 3
- Guided By : Prof Nagesh Jadhav Sir

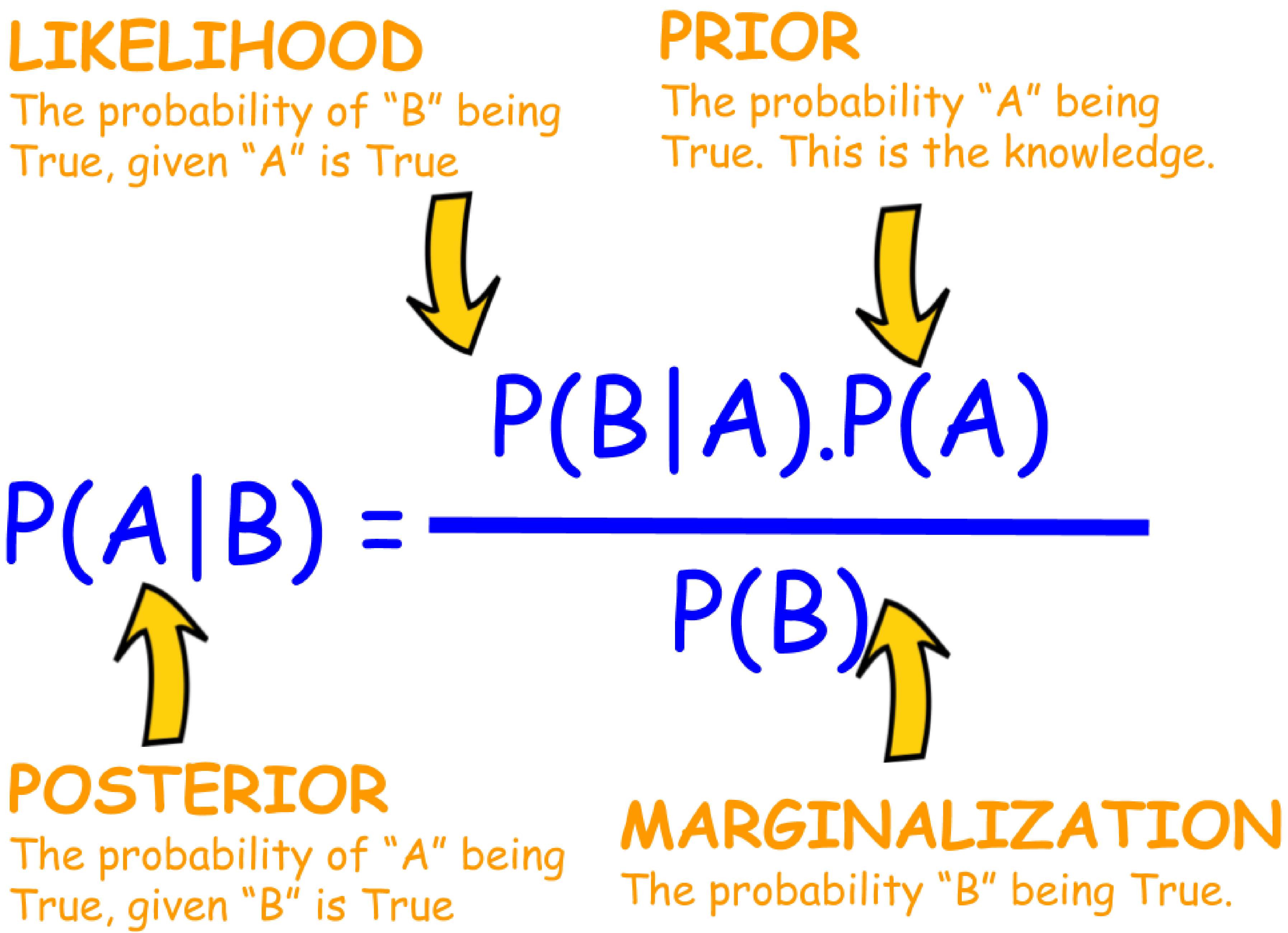
Title:- Develop a Bayesian classifier any dataset dataset

Objectives:-
1. To learn bayes theorem
2. To implement Bayesian classifier

Theory:

Bayes Theorem

- Bayes Theorem is a method to determine conditional probabilities – that is, the probability of one event occurring given that another event has already occurred. Because a conditional probability includes additional conditions – in other words, more data – it can contribute to more accurate results.
- Bayes' theorem is also known as Bayes' rule, Bayes' law, or Bayesian reasoning, which determines the probability of an event with uncertain knowledge



Types of Naive Bayes algorithm

There are 3 types of Naïve Bayes algorithm. The 3 types are listed below:-

1. Gaussian Naïve Bayes
2. Multinomial Naïve Bayes
3. Bernoulli Naïve Bayes

Applications of Naive Bayes algorithm

Naive Bayes is one of the most straightforward and fast classification algorithms. It is very well suited for large volumes of data. It is successfully used in various applications such as :

- Spam filtering
- Text classification
- Sentiment analysis
- Recommender systems It uses the Bayes theorem of probability for the prediction of unknown classes.

Bayes Classifier with example

- In machine learning, naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features.
- They are among the simplest Bayesian network models But they could be coupled with Kernel density estimation and achieve higher accuracy levels.

Naive Bayes is a simple technique for constructing classifiers:

- models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set.
- There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.
- For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

Problem Statement - Prediction of Heart cancer with the helps of medical attributes and Machine Learning algorith.

Dataset Name - Heart Disease Prediction Dataset.Kaggle

Dataset Information :

The dataset contains 76 features from 303 patients, however, published studies chose only 14 features that are relevant in predicting heart disease. Hence, In this project we will be using the dataset consisting of 303 patients with 14 features set.

Features Explanations:

- 1) age (Age in years)
- 2) sex : (1 = male, 0 = female)
- 3) cp (Chest Pain Type): [0: asymptomatic, 1: atypical angina, 2: non-anginal pain, 3: typical angina]
- 4) trestbps (Resting Blood Pressure in mm/hg)
- 5) chol (Serum Cholesterol in mg/dl)
- 6) fps (Fasting Blood Sugar > 120 mg/dl): [0 = no, 1 = yes]
- 7) restecg (Resting ECG): [0: showing probable or definite left ventricular hypertrophy by Estes' criteria, 1: normal, 2: having ST-T wave abnormality]
- 8) thalach (maximum heart rate achieved)
- 9) exang (Exercise Induced Angina): [1 = yes, 0 = no]
- 10) oldpeak (ST depression induced by exercise relative to rest)
- 11) slope (the slope of the peak exercise ST segment): [0: downsloping; 1: flat; 2: upsloping]
- 12) ca [number of major vessels (0–3)
- 13) thal : [1 = normal, 2 = fixed defect, 3 = reversible defect]
- 14) target: [0 = disease, 1 = no disease] ---> In The dataset we have 303 rows with 14 variables

  variables types:

- 1) Binary: sex, fbs, exang, target
- 2) Categorical: cp, restecg, slope, ca, thal
- 3) Continuous: age, trestbps, chol, thalac, oldpea

Import Necessary Libraries

In [1]:

```
import pandas as pd
import numpy as np

import seaborn as sns
```

```python
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

In [37]:

```python
from scipy.stats import norm
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import roc_curve, auc, confusion_matrix, classification_report, accuracy_score,roc_auc_score
from sklearn import metrics
```

In [3]:

```python
from scipy import stats
```

In [4]:

```python
from sklearn.naive_bayes import GaussianNB
```

Read Dataset

In [5]:

```python
df =pd.read_csv('heart.csv')
```

View Top 5 Rows

In [6]:

```python
df.head()
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

View Last 5 rows

In [7]:

```
df.head()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Dimensions of the Dataset

In [8]:

```
df.shape
```

Out[8]:

```
(303, 14)
```

This dataset contains 303 rows and 14 columns

Columns in dataset

In [9]:

```
df.columns
```

Out[9]:

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

Concise Summary

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

From above observation we can say that no null values in the dataset ,but lets varify by using pandas functions

Check Missing values

In [11]:

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

Out[11]:

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Check Duplicate Value

In [12]:
```
df[df.duplicated()]
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1

One duplicate row ,Just Drop That duplicate row

In [13]:
```
df.drop(df[df.duplicated()].index,axis=0,inplace=True)
```

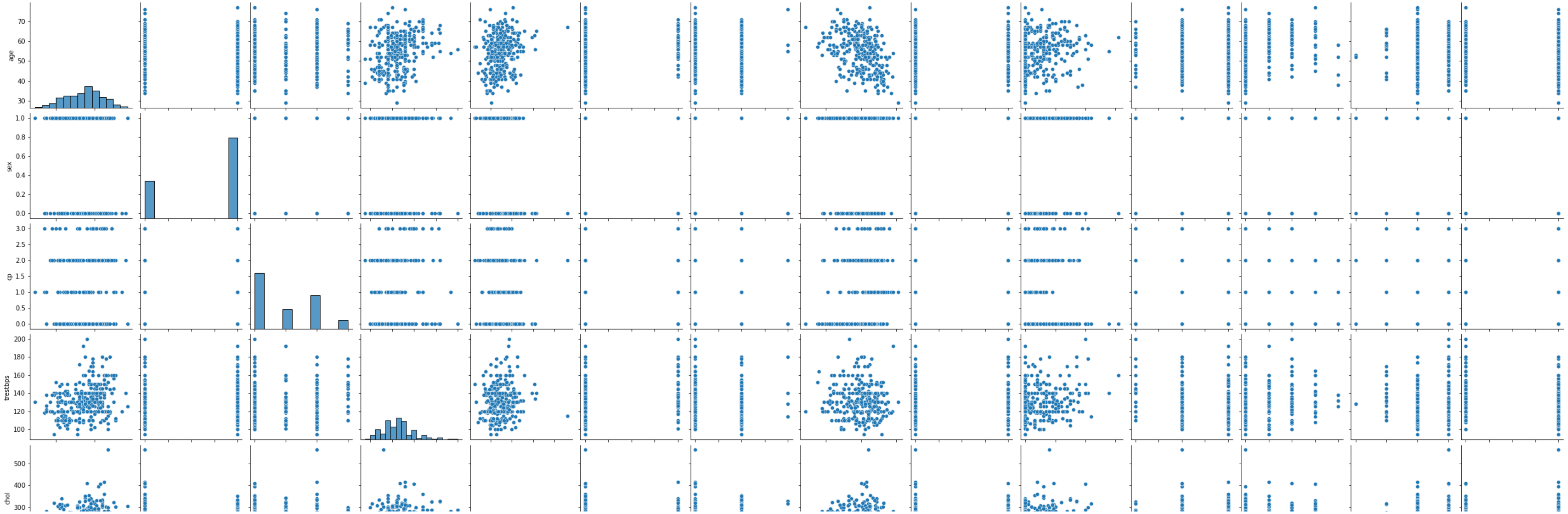
In [14]:
```
df.shape
```

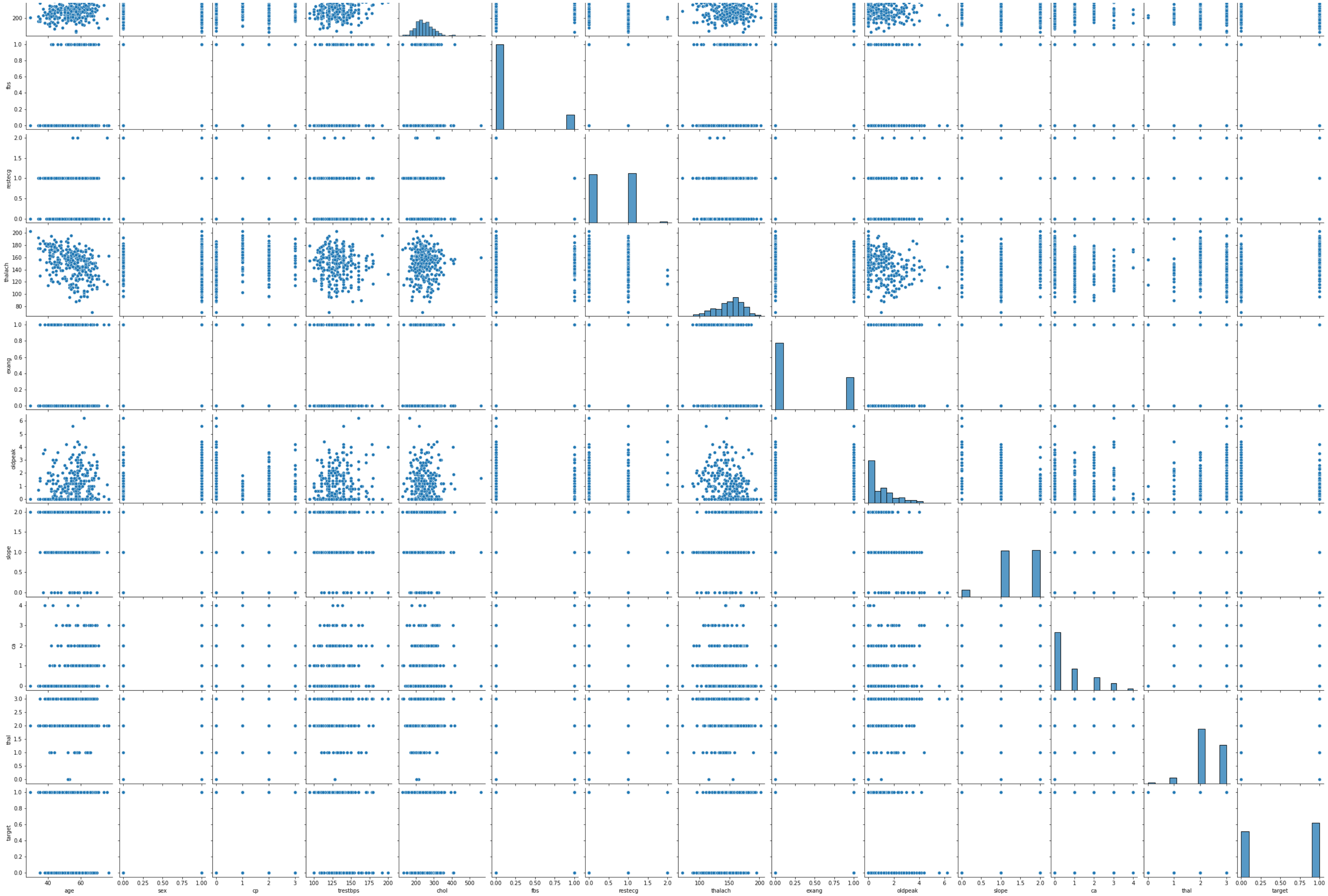
Out[14]: (302, 14)

View Relationship between Data point in dataset

In [15]:
```
sns.pairplot(data= df)
```

Out[15]: <seaborn.axisgrid.PairGrid at 0x2dde0fcfcd0>





From above figure we can say that most of datapoints are not linearly co- related with each other

Statistical Summary of data

In [16]:
```
df.describe()
```

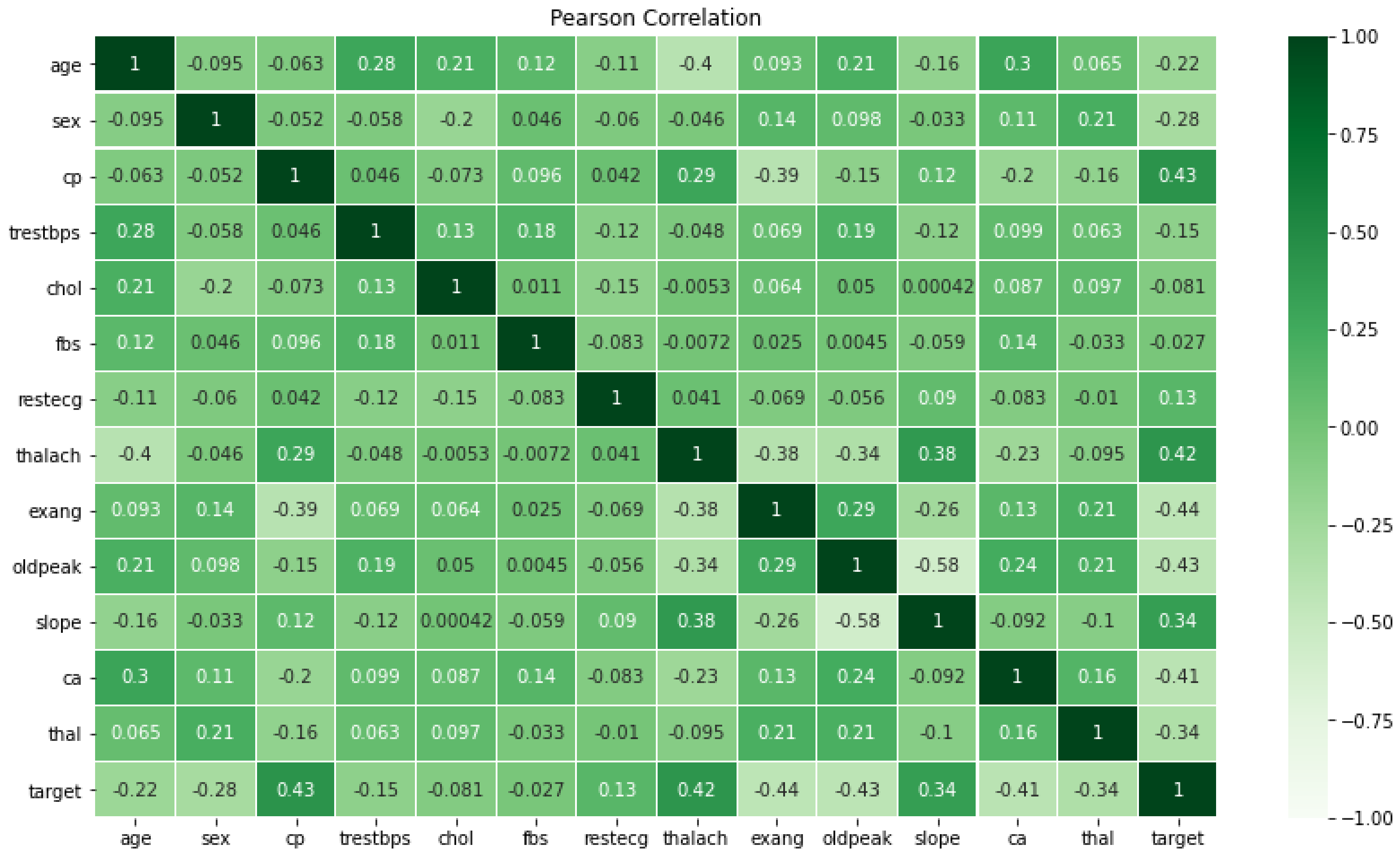
Out[16]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	149.569536	0.327815	1.043046	1.397351	0.718543	2
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	0.526027	22.903527	0.470196	1.161452	0.616274	1.006748	0
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.250000	0.000000	0.000000	1.000000	0.000000	2
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	1.000000	152.500000	0.000000	0.800000	1.000000	0.000000	2
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3

Finding the correlation between variables

In [17]:
```
pearsonCorr = df.corr(method='pearson')
spearmanCorr = df.corr(method='spearman')
fig = plt.subplots(figsize=(14,8))
sns.heatmap(pearsonCorr, vmin=-1,vmax=1, cmap = "Greens", annot=True, linewidth=0.1)
plt.title("Pearson Correlation")
```

Out[17]:
```
Text(0.5, 1.0, 'Pearson Correlation')
```

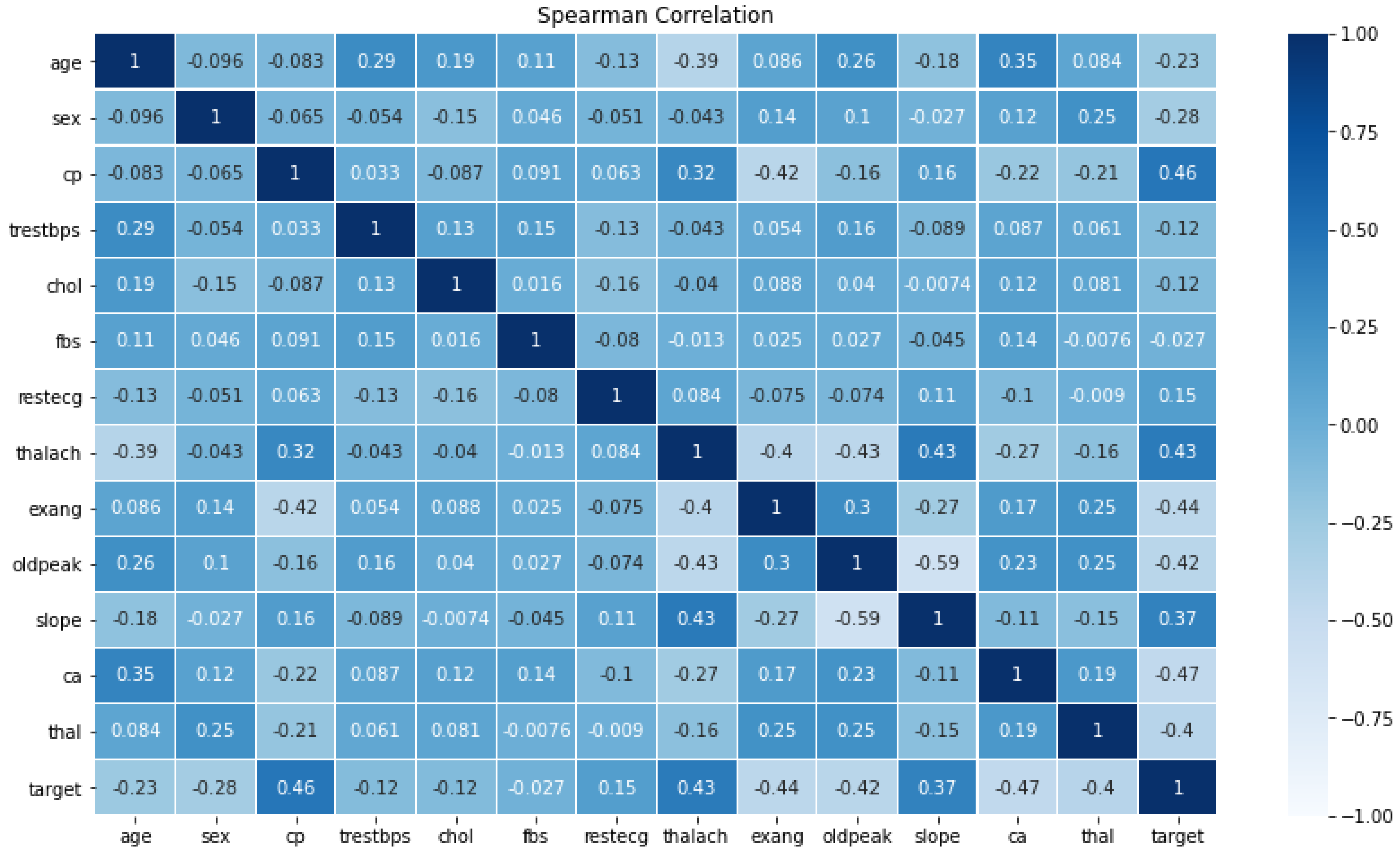


Cp is highly correlated with target
oldpeak and slope are less related with target feature

In [18]:

```
fig = plt.subplots(figsize=(14,8))
sns.heatmap(spearmanCorr, vmin=-1,vmax=1, cmap = "Blues", annot=True, linewidth=0.1)
plt.title("Spearman Correlation")
```

Out[18]: `Text(0.5, 1.0, 'Spearman Correlation')`



Cp is highly correlated with target
oldpeak and slope are less related with target feature

Check the outliers from the dataset

In [19]:

```
for i in df.columns[1:-1]:
    Q1,Q3 = np.quantile(df[i],[0.25,0.75])
    IQR = Q3-Q1
    lower_bound = Q1 - (1.5*IQR)
    if lower_bound < 0:  # as in whole dataset there is no -ve value hence we set lowerbound to 0
        lower_bound=0
    upper_bound = Q3 + (1.5*IQR)
```

```
    print(i)
    print('Lower Bound =',np.round(lower_bound,2),' Upper Bound =',np.round(upper_bound,2))
    print('min value =',df[i].min(), ' max value =', df[i].max())

    if df[i].min() < lower_bound:
        print('negative Outliers',len(df[(df[i]<lower_bound)]))

    if df[i].max() > upper_bound:
        print('positive Outliers', len(df[(df[i]>upper_bound)]))




    print('='*50)
```

```
sex
Lower Bound = 0  Upper Bound = 2.5
min value = 0  max value = 1
==================================================
cp
Lower Bound = 0  Upper Bound = 5.0
min value = 0  max value = 3
==================================================
trestbps
Lower Bound = 90.0  Upper Bound = 170.0
min value = 94  max value = 200
positive Outliers 9
==================================================
chol
Lower Bound = 115.38  Upper Bound = 370.38
min value = 126  max value = 564
positive Outliers 5
==================================================
fbs
Lower Bound = 0.0  Upper Bound = 0.0
min value = 0  max value = 1
positive Outliers 45
==================================================
restecg
Lower Bound = 0  Upper Bound = 2.5
min value = 0  max value = 2
==================================================
thalach
Lower Bound = 84.12  Upper Bound = 215.12
min value = 71  max value = 202
```

```
negative Outliers 1
==================================================
exang
Lower Bound = 0  Upper Bound = 2.5
min value = 0  max value = 1
==================================================
oldpeak
Lower Bound = 0  Upper Bound = 4.0
min value = 0.0  max value = 6.2
positive Outliers 5
==================================================
slope
Lower Bound = 0  Upper Bound = 3.5
min value = 0  max value = 2
==================================================
ca
Lower Bound = 0  Upper Bound = 2.5
min value = 0  max value = 4
positive Outliers 24
==================================================
thal
Lower Bound = 0.5  Upper Bound = 4.5
min value = 0  max value = 3
negative Outliers 2
==================================================
```

From above result we can say that fbs feature contains higher positive outlier

Drop the outlier using Z score

In [20]:

```
z = np.abs(stats.zscore(df))
df = df[(z<3).all(axis=1)]
df.shape
```

Out[20]:

```
(287, 14)
```

after Droping outlier we are left with 287 rows and 14 columns

In [21]:

```
df.target.unique()
```

Out[21]:

```
array([1, 0], dtype=int64)
```

Value count of target variable

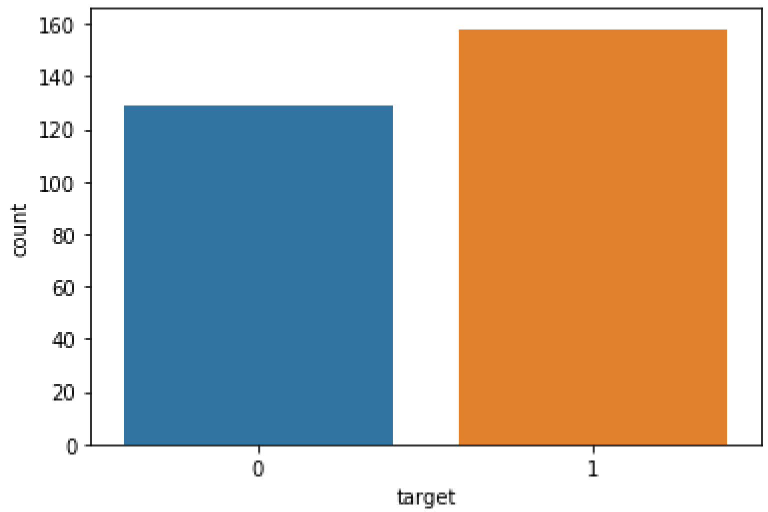
In [22]:
```
df.target.value_counts()
```

Out[22]:
```
1    158
0    129
Name: target, dtype: int64
```

Plot the value count

In [23]:
```
sns.countplot('target',data=df,label ="Target")
```

Out[23]: <AxesSubplot:xlabel='target', ylabel='count'>



Model Building

In [26]:
```
X =df.drop(['target'], axis =1)
Y = df['target']
```

In [27]:
```
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_state=42)
```

In [28]:
```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```
Out[28]:
```
((229, 13), (58, 13), (229,), (58,))
```
In [25]:
```
scaler = StandardScaler()
```
In [30]:
```
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```
In [31]:
```
GNB = GaussianNB()
```
In [32]:
```
GNB.fit(X_train,y_train)
```
Out[32]:
```
GaussianNB()
```
In [33]:
```
GNB.score(X_train,y_train)
```
Out[33]:
```
0.8646288209606987
```
Training Accuracy is 87%

In [34]:
```
GNB.score(X_test,y_test)
```
Out[34]:
```
0.8103448275862069
```
Testing Accuracy is 82%

Predict on X_train

In [38]:
```
y_pred_train = GNB.predict(X_train)
Accuracy_train=metrics.accuracy_score(y_train,y_pred_train )
precision_train=metrics.precision_score(y_train,y_pred_train)
recall_train=metrics.recall_score(y_train,y_pred_train)
```

```
f1_score_train=metrics.f1_score(y_train,y_pred_train)
roc_auc_train=metrics.roc_auc_score(y_train,y_pred_train)
```

In [39]:

```
print("Model Name = GaussianNB")
print("Accuracy is =",Accuracy_train)
print("Precision score is =",precision_train)
print("Recall score = ",recall_train)
print("f1 SCore score is = ",f1_score_train)
print("Roc_Auc score is= ",roc_auc_train)
```

```
Model Name = GaussianNB
Accuracy is = 0.8646288209606987
Precision score is = 0.8650793650793651
Recall score =  0.8861788617886179
f1 SCore score is =  0.8755020080321285
Roc_Auc score is=  0.862900751649026
```

Predict on X_test

In [40]:

```
y_pred_test = GNB.predict(X_test)
Accuracy_test=metrics.accuracy_score(y_test,y_pred_test)
precision_test=metrics.precision_score(y_test,y_pred_test)
recall_test=metrics.recall_score(y_test,y_pred_test)
f1_score_test=metrics.f1_score(y_test,y_pred_test)
roc_auc_test=metrics.roc_auc_score(y_test,y_pred_test)
```

In [42]:

```
print("Model Name = GaussianNB")
print("Accuracy is ",Accuracy_test)
print("Precision score is ",precision_test)
print("Recall _score is",recall_test)
print("f1 SCore score is ",f1_score_test)
print("Roc_Auc score is",roc_auc_test)
```

```
Model Name = GaussianNB
Accuracy is  0.8103448275862069
Precision score is  0.875
Recall _score is 0.8
f1 SCore score is  0.8358208955223881
Roc_Auc score is 0.8130434782608696
```

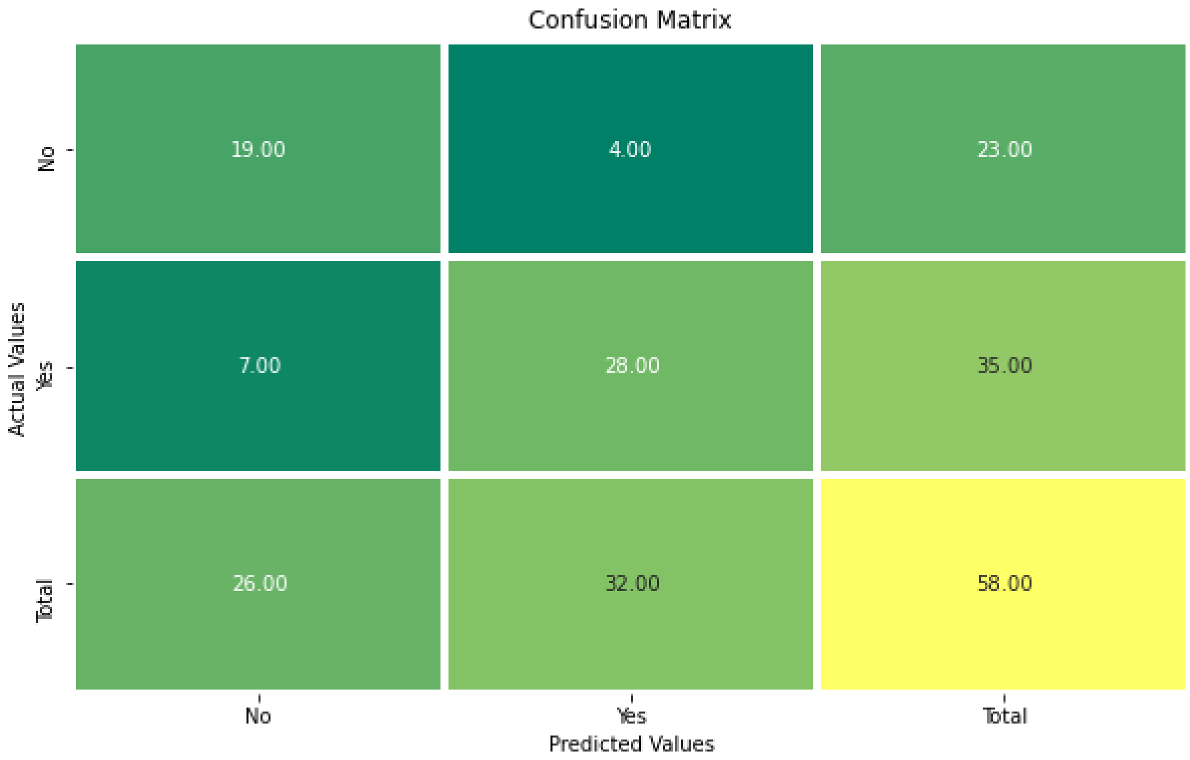
Confusion Matrix

In [47]:

```
label=['No','Yes']
```

In [48]:

```
# confusion matrix
cm = confusion_matrix(y_test,y_pred_test)
row_sum = cm.sum(axis=0)
cm = np.append(cm,row_sum.reshape(1,-1),axis=0)
col_sum = cm.sum(axis=1)
cm = np.append(cm,col_sum.reshape(-1,1),axis=1)
labels = label+['Total']
plt.figure(figsize=(10,6))
sns.heatmap(cm,annot=True,cmap='summer',fmt='0.2f',xticklabels=labels,
yticklabels=labels,linewidths=3,cbar=None,)
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Confusion Matrix')
plt.show()
```



Classification Report

In [49]:

```
print('*'*30+'Classifcation Report'+'*'*30+'\n\n')
cr = classification_report(y_test,y_pred)
print(cr)
```

```
******************************Classifcation Report******************************


              precision    recall  f1-score   support

           0       0.73      0.83      0.78        23
           1       0.88      0.80      0.84        35

    accuracy                           0.81        58
   macro avg       0.80      0.81      0.81        58
weighted avg       0.82      0.81      0.81        58
```

Conclusion

Thus we have successfully completed the implementation of Naïve Bayes Gaussian Classifier

In [ ]: