In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score,recall_score,accuracy_scor
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier,AdaBoostClassifier,Ranc
```

In [3]:
```
df=pd.read_csv("C:/Users/dell/Downloads/heart.csv")
```

In [4]:
```
df.head()           #It Shows First 5 Rows of Dataset
```

Out[4]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Exe
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 [5]:
```
df.shape           #It Shows Count of Rows & Column Present in Datas
```

Out[5]:
```
(918, 12)
```

In [6]:
```
df.size           #It Shows Total Number of Element Present in Datas
```

Out[6]:
```
11016
```

In [7]:
```
df.columns           #It Shows All Column Names
```

Out[7]:
```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'Fastin
gBS',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
```

In [8]:
```
df.info()          #It Shows Overall Discription of Dataset
```

```
<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 [9]:
```
df.isnull().sum()          #Checking Null Values Present in Dataset
```

Out[9]:
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

In [10]:
```
df.dtypes                  #It Shows DataTypes of All Columns
```

Out[10]:
```
Age                 int64
Sex                object
ChestPainType      object
RestingBP           int64
Cholesterol         int64
FastingBS           int64
RestingECG         object
MaxHR               int64
ExerciseAngina     object
Oldpeak           float64
ST_Slope           object
HeartDisease        int64
dtype: object
```

Label-Encoding Treatment :

-Machines Can't Understand Catagorical Data That's Why We Use Label Encoding.

-Label Encoding is Used For Coverting Catagorical Data into Numerical Value.

In [11]:

```
le=LabelEncoder()
```

In [12]:

```
df["Sex"]=le.fit_transform(df["Sex"])
df["ChestPainType"]=le.fit_transform(df["ChestPainType"])
df["RestingECG"]=le.fit_transform(df["RestingECG"])
df["ExerciseAngina"]=le.fit_transform(df["ExerciseAngina"])
df["ST_Slope"]=le.fit_transform(df["ST_Slope"])
```

In [13]:

```
df.dtypes
```

Out[13]:

```
Age                 int64
Sex                 int32
ChestPainType       int32
RestingBP           int64
Cholesterol         int64
FastingBS           int64
RestingECG          int32
MaxHR               int64
ExerciseAngina      int32
Oldpeak           float64
ST_Slope            int32
HeartDisease        int64
dtype: object
```

In [14]:

```
df.head()
```

Out[14]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Exe
0	40	1	1	140	289	0	1	172	
1	49	0	2	160	180	0	1	156	
2	37	1	1	130	283	0	2	98	
3	48	0	0	138	214	0	1	108	
4	54	1	2	150	195	0	1	122	

Changing DataTypes :

-It is Used For Coverting One DataType into Another Datatype.

In [15]:

```
df["Oldpeak"]=df["Oldpeak"].astype("int")
```

In [16]:

```
df.head()
```

Out[16]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Exe
0	40	1	1	140	289	0	1	172	
1	49	0	2	160	180	0	1	156	
2	37	1	1	130	283	0	2	98	
3	48	0	0	138	214	0	1	108	
4	54	1	2	150	195	0	1	122	

Statistical Summary :

-It Shows 5 Point Summary of Dataset.

In [17]:

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

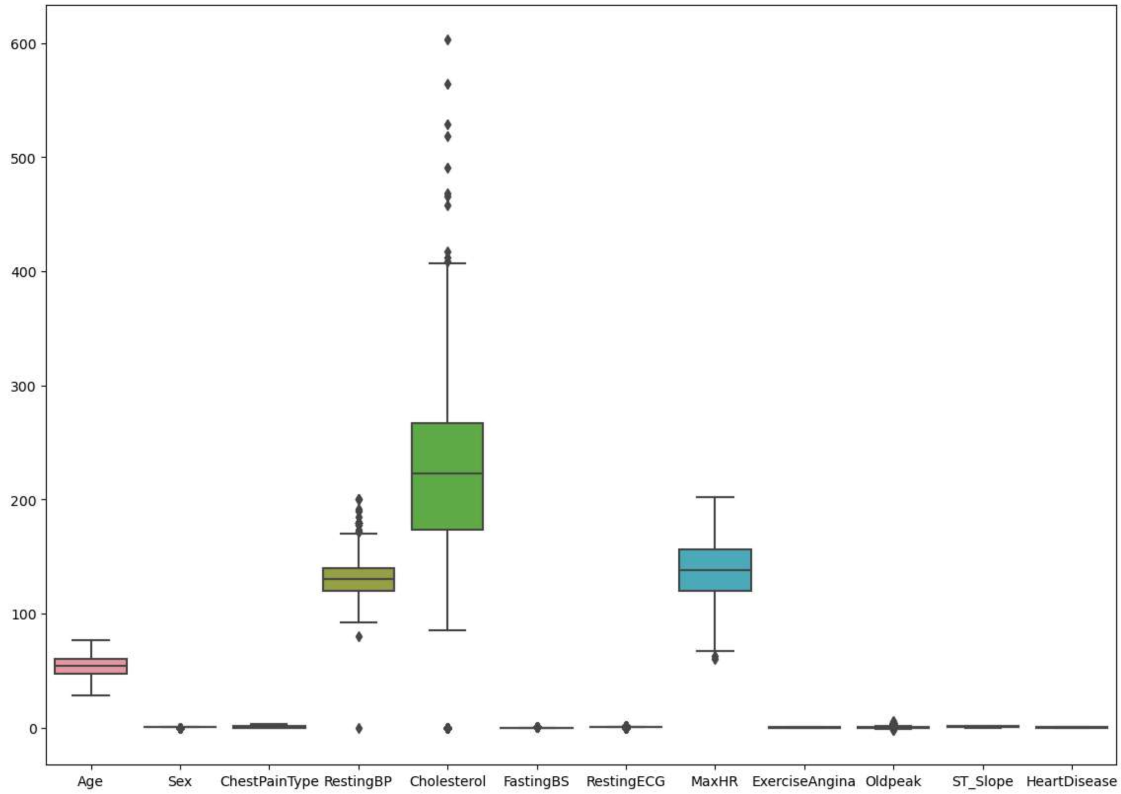
Out[17]:

	count	mean	std	min	25%	50%	75%	max
Age	918.0	53.510893	9.432617	28.0	47.00	54.0	60.0	77.0
Sex	918.0	0.789760	0.407701	0.0	1.00	1.0	1.0	1.0
ChestPainType	918.0	0.781046	0.956519	0.0	0.00	0.0	2.0	3.0
RestingBP	918.0	132.396514	18.514154	0.0	120.00	130.0	140.0	200.0
Cholesterol	918.0	198.799564	109.384145	0.0	173.25	223.0	267.0	603.0
FastingBS	918.0	0.233115	0.423046	0.0	0.00	0.0	0.0	1.0
RestingECG	918.0	0.989107	0.631671	0.0	1.00	1.0	1.0	2.0
MaxHR	918.0	136.809368	25.460334	60.0	120.00	138.0	156.0	202.0
ExerciseAngina	918.0	0.404139	0.490992	0.0	0.00	0.0	1.0	1.0
Oldpeak	918.0	0.720044	0.990165	-2.0	0.00	0.0	1.0	6.0
ST_Slope	918.0	1.361656	0.607056	0.0	1.00	1.0	2.0	2.0
HeartDisease	918.0	0.553377	0.497414	0.0	0.00	1.0	1.0	1.0

Outliers Treatment :

In [18]:

```
plt.figure(figsize=(14,10))
sns.boxplot(df)
plt.show()
```



Steps of Removing Outliers :

In [19]:

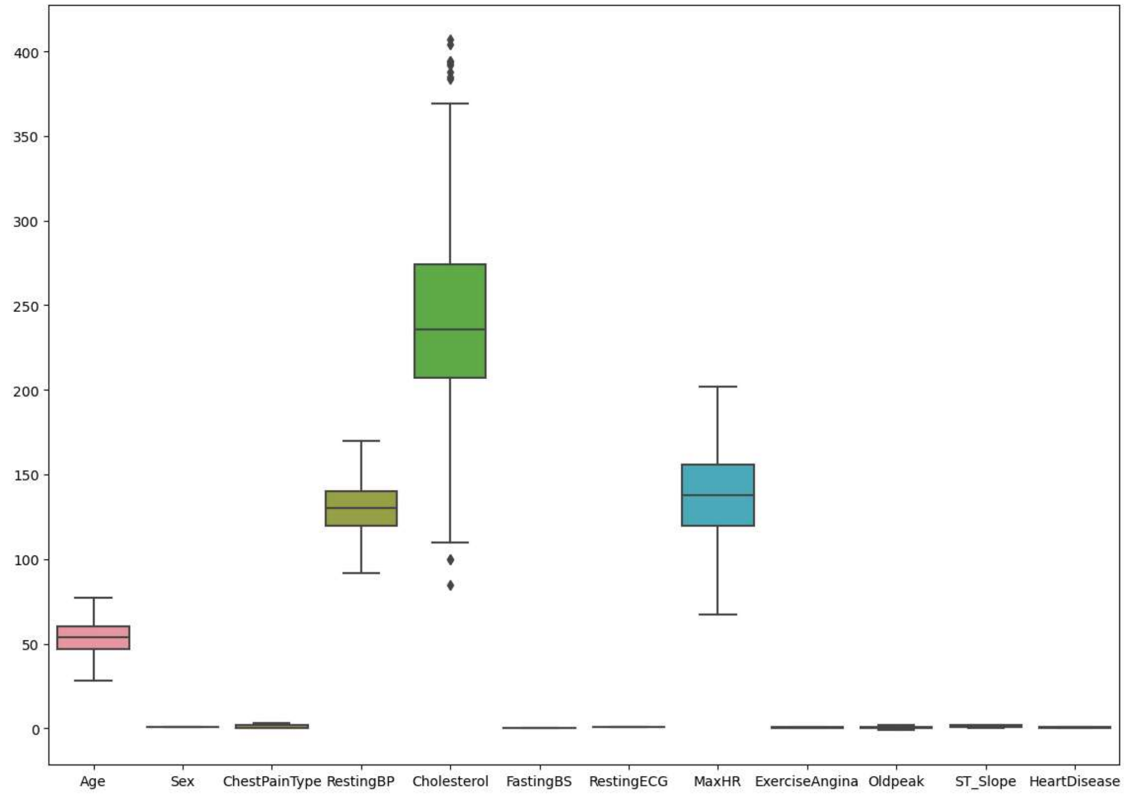
```
Q1=df.quantile(q=0.25)                  #Finding Q1 Value
Q3=df.quantile(q=0.75)                  #Finding Q3 Value
IQR=Q3-Q1                               #Finding IQR Value i.e.(Inter Qu
upper=Q3+(1.5*IQR)                      #To Detect Upper Outliers
lower=Q1-(1.5*IQR)                      #To Detect Lower Outliers
```

In [20]:

```
df1=df[~((df>upper) | (df<lower))]     #Remove the data which is less t
```

Box Plot After Removing Outliers :

In [21]:
```
plt.figure(figsize=(14,10))
sns.boxplot(df1)
plt.show()
```



In [22]:
```
df1.isnull().sum()                    #Checking Null Values After Remov
```

Out[22]:
```
Age                 0
Sex               193
ChestPainType       0
RestingBP          28
Cholesterol       183
FastingBS         214
RestingECG        366
MaxHR               2
ExerciseAngina      0
Oldpeak            58
ST_Slope            0
HeartDisease        0
dtype: int64
```

In [23]:
```
df2=df1.dropna()                        #Dropping Null Values
```

In [24]:
```
df2.isnull().sum()
```

Out[24]:
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

Heatmap :

-Heatmap Shows Two Dimensional Graphical Representation of Data.

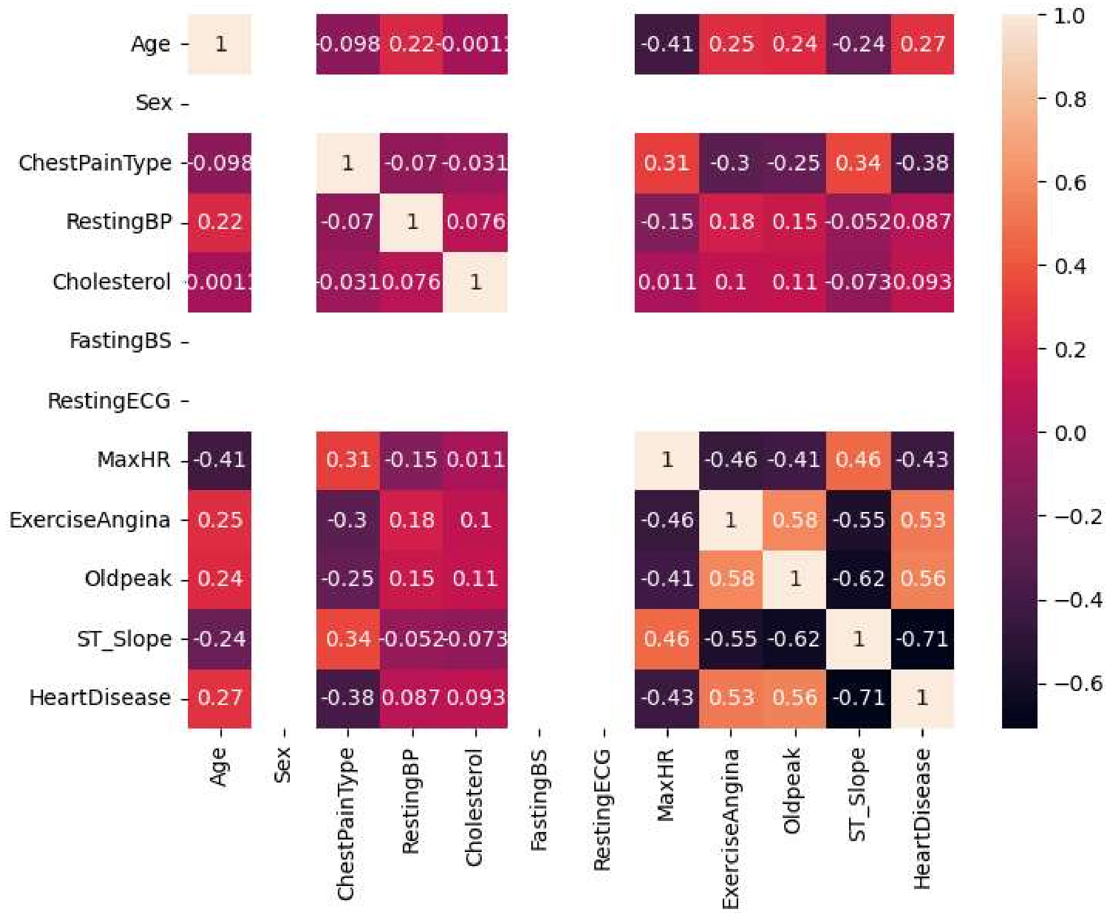
In [25]:
```
df2.corr()                      #Finding Pairwise Correlation of All (
```

Out[25]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	Resting
Age	1.000000	NaN	-0.097992	0.222260	-0.001110	NaN	
Sex	NaN	NaN	NaN	NaN	NaN	NaN	
ChestPainType	-0.097992	NaN	1.000000	-0.069858	-0.031305	NaN	
RestingBP	0.222260	NaN	-0.069858	1.000000	0.076330	NaN	
Cholesterol	-0.001110	NaN	-0.031305	0.076330	1.000000	NaN	
FastingBS	NaN	NaN	NaN	NaN	NaN	NaN	
RestingECG	NaN	NaN	NaN	NaN	NaN	NaN	
MaxHR	-0.411458	NaN	0.314437	-0.146843	0.011218	NaN	
ExerciseAngina	0.246757	NaN	-0.301003	0.180704	0.103847	NaN	
Oldpeak	0.244543	NaN	-0.250545	0.146872	0.114248	NaN	
ST_Slope	-0.241619	NaN	0.343479	-0.051887	-0.073499	NaN	
HeartDisease	0.267243	NaN	-0.377125	0.087065	0.092862	NaN	

In [26]:

```
plt.figure(figsize=(8,6))                                   #Range -1 to 1
sns.heatmap(df2.corr(),annot=True)                          #Correlation between
plt.show()
```



Pair-Plot :

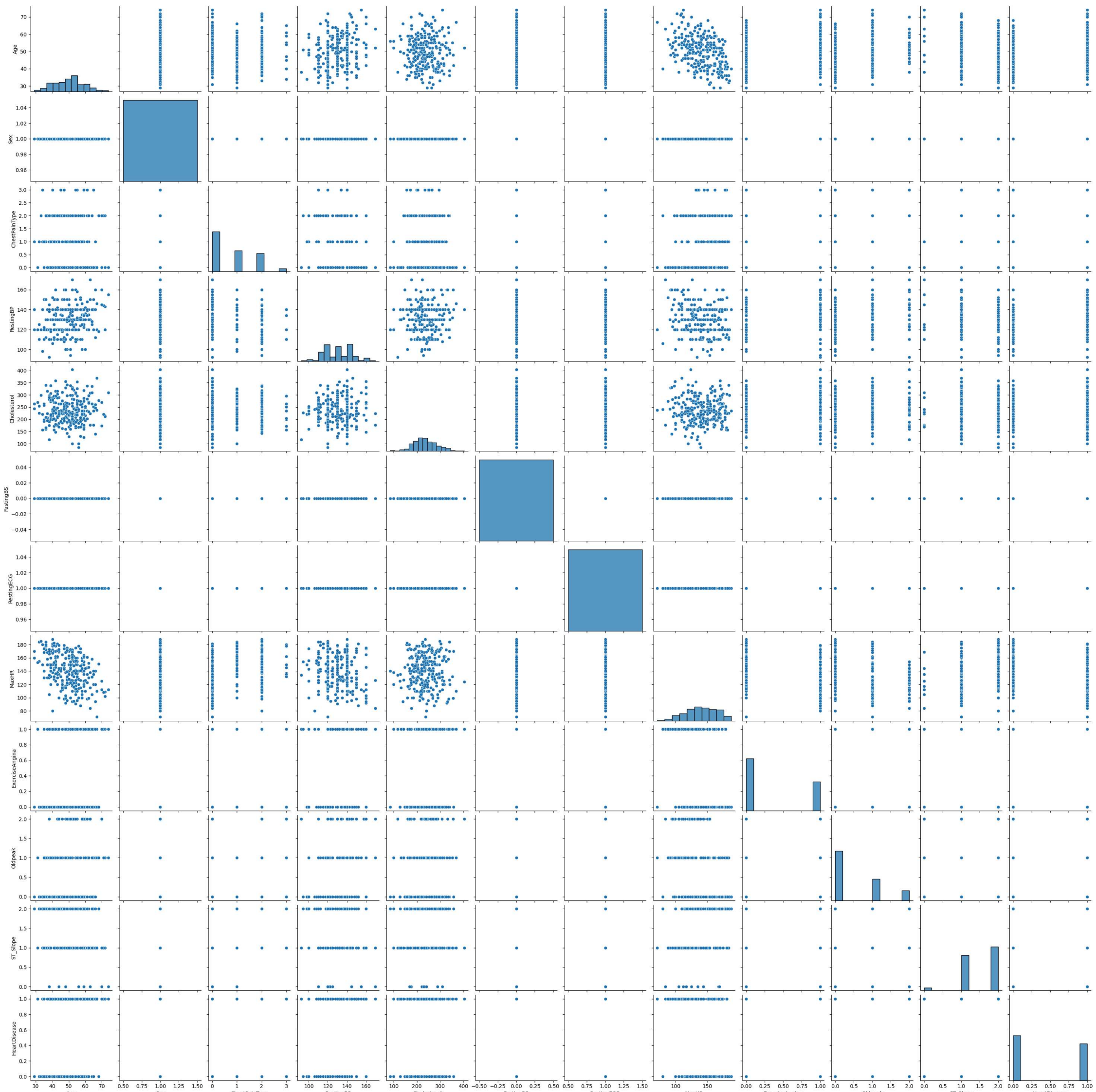
-It is Used Plot Multiple Pairwise Distributions in Dataset.

In [70]:

```
sns.pairplot(df2)
```

```
C:\Users\dell\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWa
rning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[70]: <seaborn.axisgrid.PairGrid at 0x1cfd565d050>



Pie Chart :

-It is a Proportional Representation of the Data in a Column.

In [27]:

```
df["HeartDisease"].unique()
```

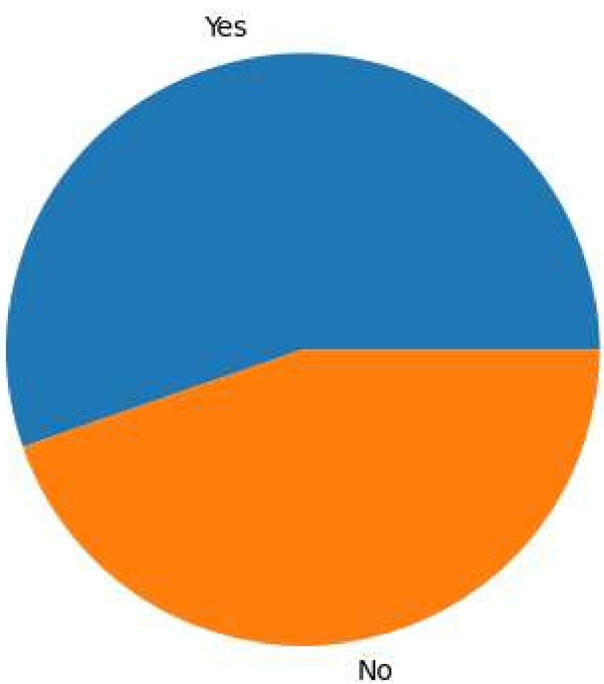
Out[27]: array([0, 1], dtype=int64)

In [28]:
```
df["HeartDisease"].value_counts(normalize=True)*100
```

Out[28]:
```
HeartDisease
1    55.337691
0    44.662309
Name: proportion, dtype: float64
```

In [29]:
```
plt.pie(df["HeartDisease"].value_counts(normalize=True)*100,labels=["\
```

Out[29]:
```
([<matplotlib.patches.Wedge at 0x16fa3ba1fd0>,
  <matplotlib.patches.Wedge at 0x16fa4332a90>],
 [Text(-0.1835941114214546, 1.0845705151124876, 'Yes'),
  Text(0.1835941114214547, -1.0845705151124876, 'No')])
```



MODEL BUILDING :

In [35]:
```
x=df2.drop(["HeartDisease"],axis=1)          #Segrigation of data
y=df2["HeartDisease"]                        #x is independent fea
```

In [36]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,rando
```

In [37]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(212, 11)
(53, 11)
(212,)
(53,)
```

1.LOGISTIC REGRESSION ALGORITHM :

In [38]:
```
lr=LogisticRegression()            #Object creation
lr.fit(x_train,y_train)            #Fitted on train data
```

```
C:\Users\dell\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://s
cikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression (https://scikit-learn.org/stable/modules/linear_model.html#logis
tic-regression)
  n_iter_i = _check_optimize_result(
```

Out[38]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [40]:
```
y_true,y_pred=y_test,lr.predict(x_test)    #
print(lr.score(x_train,y_train)*100)
print(lr.score(x_test,y_test)*100)
```

```
85.84905660377359
83.01886792452831
```

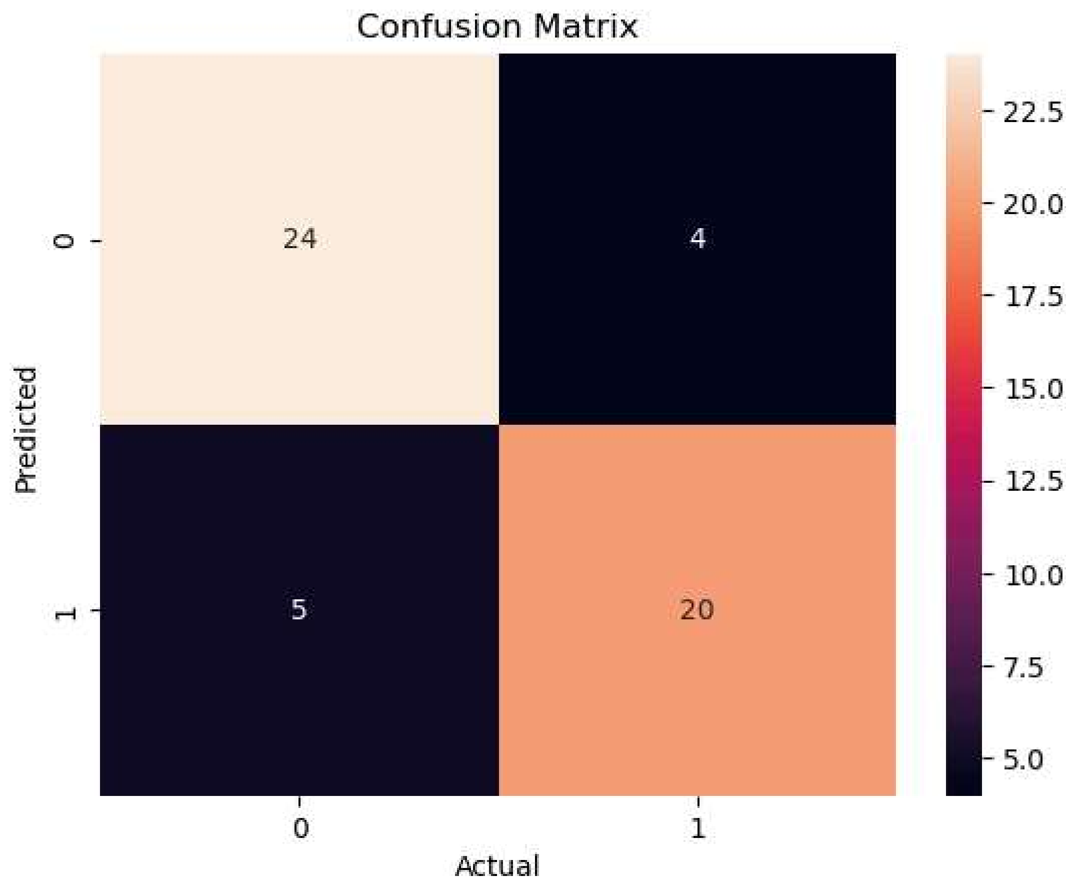
In [41]:
```
print(precision_score(y_true,y_pred)*100)        #Predicted +ve by mac
print(recall_score(y_true,y_pred)*100)           #Real +ve
print(accuracy_score(y_true,y_pred)*100)
```

```
83.33333333333334
80.0
83.01886792452831
```

In [42]:
```
print(confusion_matrix(y_true,y_pred))

sns.heatmap(confusion_matrix(y_true,y_pred),annot=True)
plt.title("Confusion Matrix")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
[[24  4]
 [ 5 20]]
```



2.K-NEAREST NEIGHBOUR CLASSIFIER ALGORITHM :

In [128]:
```
knn=KNeighborsClassifier(n_neighbors=30,weights='uniform')
knn.fit(x_train,y_train)                                   #It classify t
```

Out[128]: `KNeighborsClassifier(n_neighbors=30)`

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [129]:

```
y_true,y_pred=y_test,knn.predict(x_test)
print(knn.score(x_train,y_train)*100)
print(knn.score(x_test,y_test)*100)
```

```
71.22641509433963
69.81132075471697
```

In [130]:

```
print(precision_score(y_true,y_pred)*100)
print(recall_score(y_true,y_pred)*100)
print(accuracy_score(y_true,y_pred)*100)
```

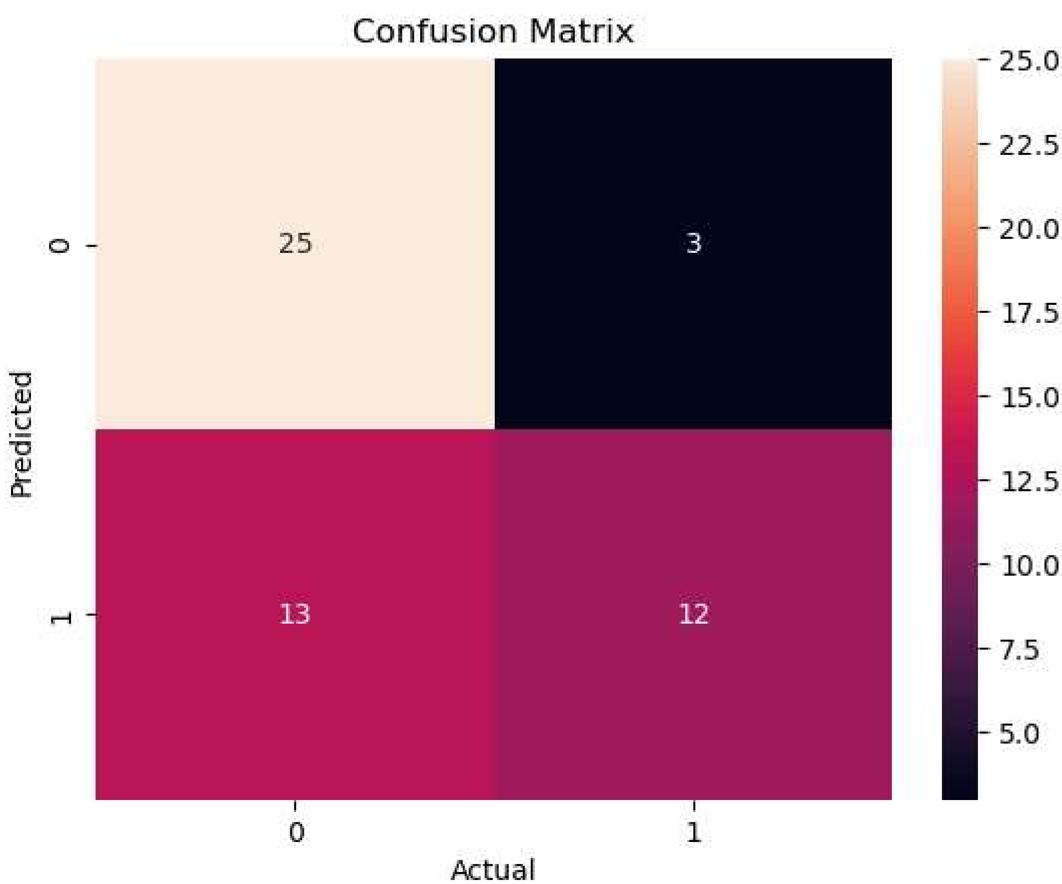
```
80.0
48.0
69.81132075471697
```

In [131]:

```
print(confusion_matrix(y_true,y_pred))

sns.heatmap(confusion_matrix(y_true,y_pred),annot=True)
plt.title("Confusion Matrix")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
[[25  3]
 [13 12]]
```



3.SUPPORT VECTOR CLASSIFIER ALGORITHM :

In [120]:

```
svc=SVC(C=1.0,kernel='linear')
svc.fit(x_train,y_train)
```

Out[120]:

```
SVC(kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [121]:

```
y_true,y_pred=y_test,svc.predict(x_test)
print(svc.score(x_train,y_train)*100)
print(svc.score(x_test,y_test)*100)
```

```
88.20754716981132
83.01886792452831
```

In [122]:

```
print(precision_score(y_true,y_pred)*100)
print(recall_score(y_true,y_pred)*100)
print(accuracy_score(y_true,y_pred)*100)
```

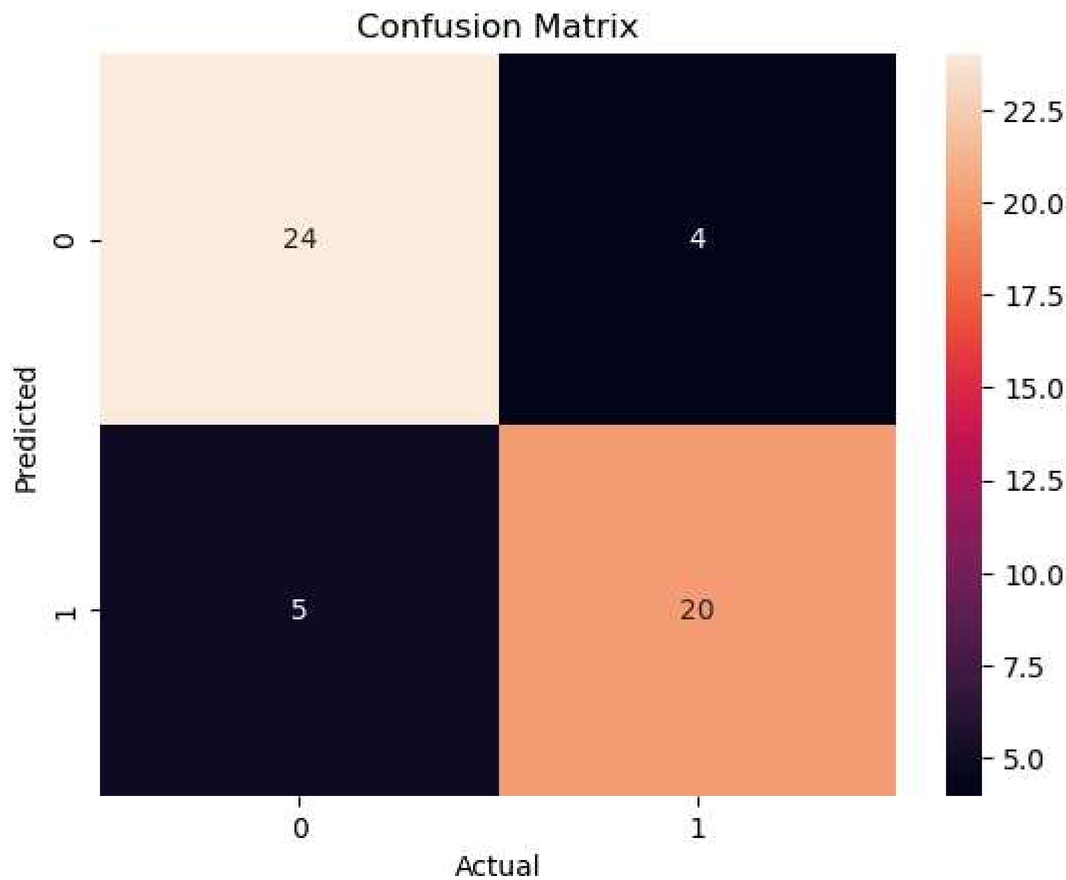
```
83.33333333333334
80.0
83.01886792452831
```

In [123]:

```
print(confusion_matrix(y_true,y_pred))

sns.heatmap(confusion_matrix(y_true,y_pred),annot=True)
plt.title("Confusion Matrix")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
[[24  4]
 [ 5 20]]
```



4.RANDOM FOREST CLASSIFIER ALGORITHM :

In [88]:

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

Out[88]:

```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [89]:

```
y_true,y_pred=y_test,rf.predict(x_test)
print(rf.score(x_train,y_train)*100)
print(rf.score(x_train,y_train)*100)
```

```
100.0
100.0
```

In [90]:

```
print(precision_score(y_true,y_pred)*100)
print(recall_score(y_true,y_pred)*100)
print(accuracy_score(y_true,y_pred)*100)
```

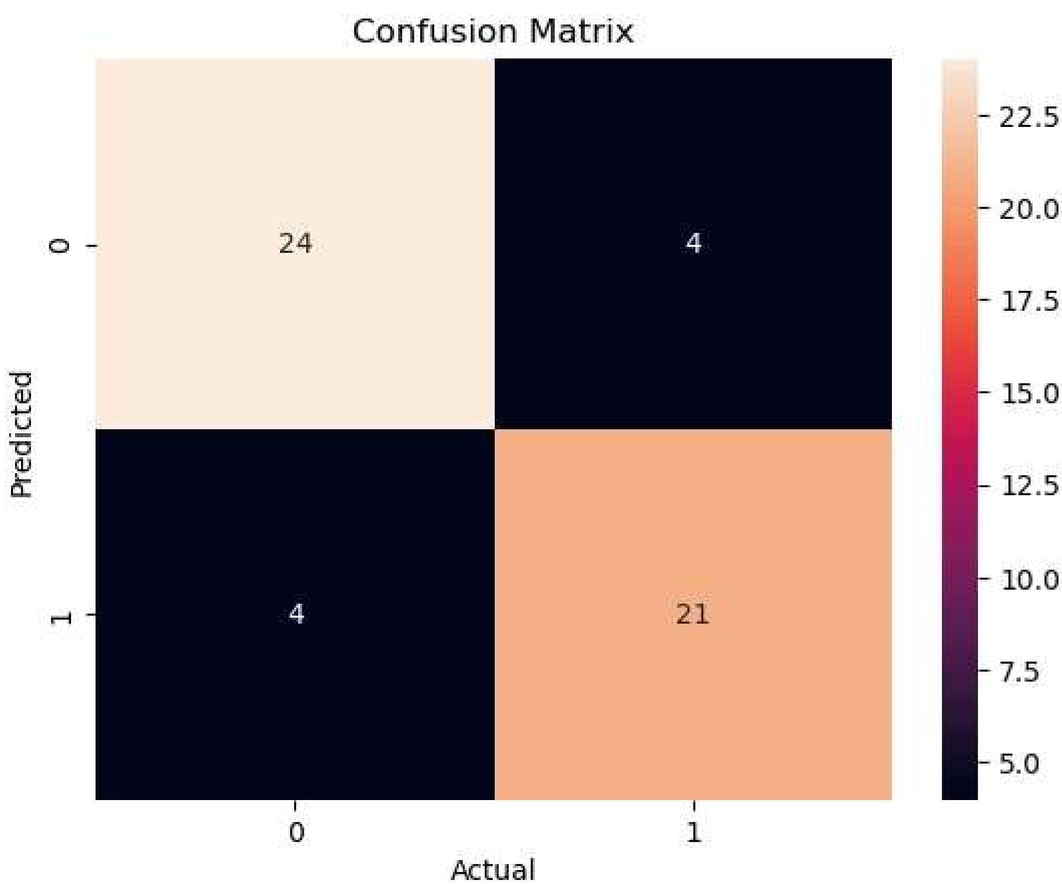
```
84.0
84.0
84.90566037735849
```

In [91]:

```
print(confusion_matrix(y_true,y_pred))

sns.heatmap(confusion_matrix(y_true,y_pred),annot=True)
plt.title("Confusion Matrix")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
[[24  4]
 [ 4 21]]
```



5.DECISION TREE CLASSIFIER ALGORITHM :

In [100]:

```
dt=DecisionTreeClassifier(criterion='gini',max_depth=3)
dt.fit(x_train,y_train)
```

Out[100]:

```
DecisionTreeClassifier(max_depth=3)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [101]:

```
y_true,y_pred=y_test,dt.predict(x_test)
print(dt.score(x_train,y_train)*100)
print(dt.score(x_test,y_test)*100)
```

```
89.62264150943396
86.79245283018868
```

In [102]:

```
print(precision_score(y_true,y_pred)*100)
print(recall_score(y_true,y_pred)*100)
print(accuracy_score(y_true,y_pred)*100)
```

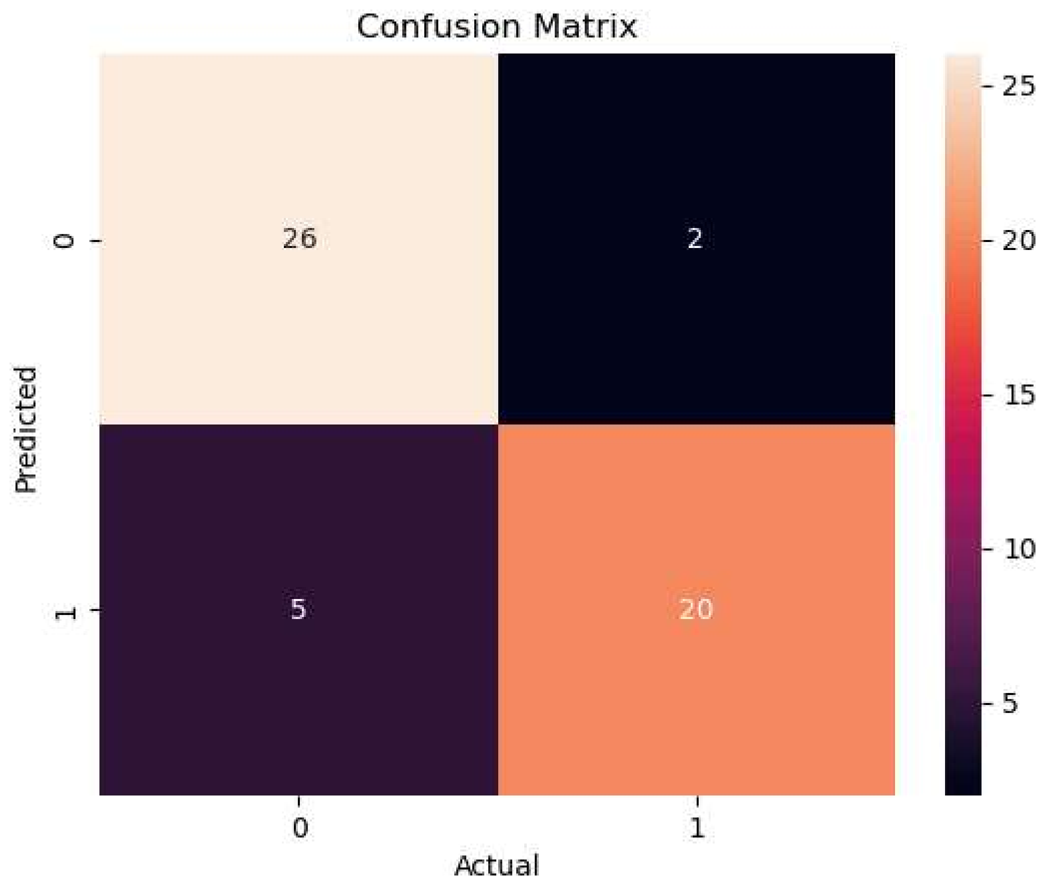
```
90.9090909090909
80.0
86.79245283018868
```

In [103]:

```
print(confusion_matrix(y_true,y_pred))

sns.heatmap(confusion_matrix(y_true,y_pred),annot=True)
plt.title("Confusion Matrix")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
[[26  2]
 [ 5 20]]
```



6.BAGGING CLASSIFIER ALGORITHM :

In [104]:

```
bg=BaggingClassifier(n_estimators=5,random_state=1)
bg.fit(x_train,y_train)
```

Out[104]:

```
BaggingClassifier(n_estimators=5, random_state=1)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [105]:
```
y_true,y_pred=y_test,bg.predict(x_test)
print(bg.score(x_train,y_train)*100)
print(bg.score(x_test,y_test)*100)
```

```
96.69811320754717
79.24528301886792
```

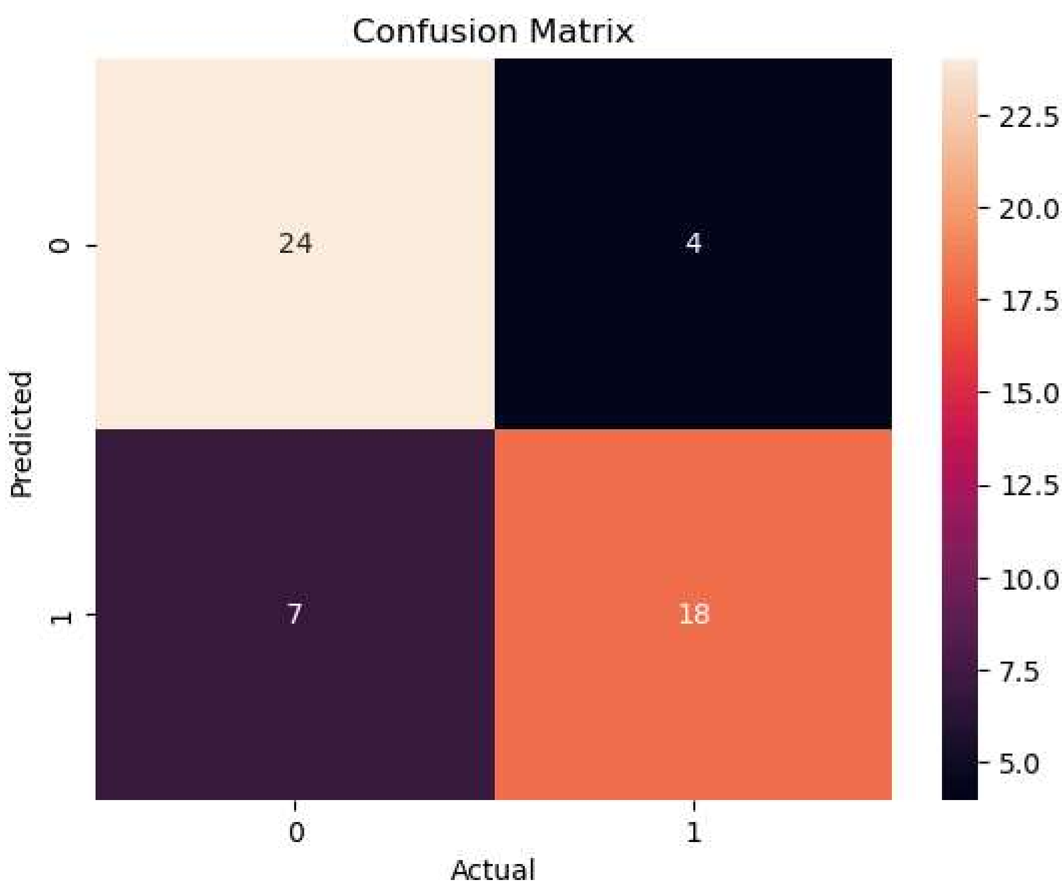
In [106]:
```
print(precision_score(y_true,y_pred)*100)
print(recall_score(y_true,y_pred)*100)
print(accuracy_score(y_true,y_pred)*100)
```

```
81.81818181818183
72.0
79.24528301886792
```

In [107]:
```
print(confusion_matrix(y_true,y_pred))

sns.heatmap(confusion_matrix(y_true,y_pred),annot=True)
plt.title("Confusion Matrix")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
[[24  4]
 [ 7 18]]
```



7.GRADINT BOOSTING CLASSIFIER ALGORITHM :

In [108]:

```
gb=GradientBoostingClassifier(n_estimators=5)
gb.fit(x_train,y_train)
```

Out[108]:

```
GradientBoostingClassifier(n_estimators=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [109]:

```
y_true,y_pred=y_test,gb.predict(x_test)
print(gb.score(x_train,y_train)*100)
print(gb.score(x_test,y_test)*100)
```

```
91.0377358490566
86.79245283018868
```

In [110]:

```
print(precision_score(y_true,y_pred)*100)
print(recall_score(y_true,y_pred)*100)
print(accuracy_score(y_true,y_pred)*100)
```

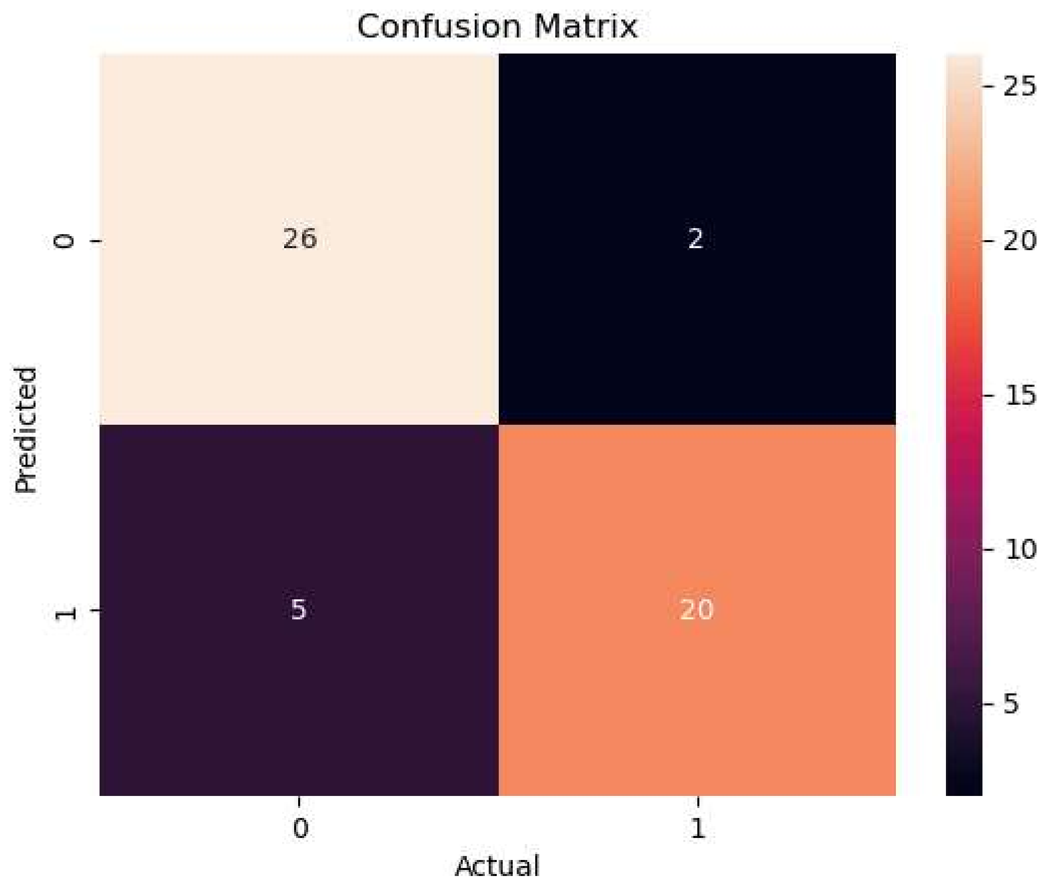
```
90.9090909090909
80.0
86.79245283018868
```

In [111]:

```
print(confusion_matrix(y_true,y_pred))

sns.heatmap(confusion_matrix(y_true,y_pred),annot=True)
plt.title("Confusion Matrix")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
[[26  2]
 [ 5 20]]
```



8.ADA-BOOST CLASSIFIER ALGORITHM :

In [112]:

```
ad=AdaBoostClassifier()
ad.fit(x_train,y_train)
```

Out[112]: AdaBoostClassifier()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [113]:

```
y_true,y_pred=y_test,ad.predict(x_test)
print(ad.score(x_train,y_train)*100)
print(ad.score(x_test,y_test)*100)
```

```
93.86792452830188
83.01886792452831
```

In [114]:

```
print(precision_score(y_true,y_pred)*100)
print(recall_score(y_true,y_pred)*100)
print(accuracy_score(y_true,y_pred)*100)
```

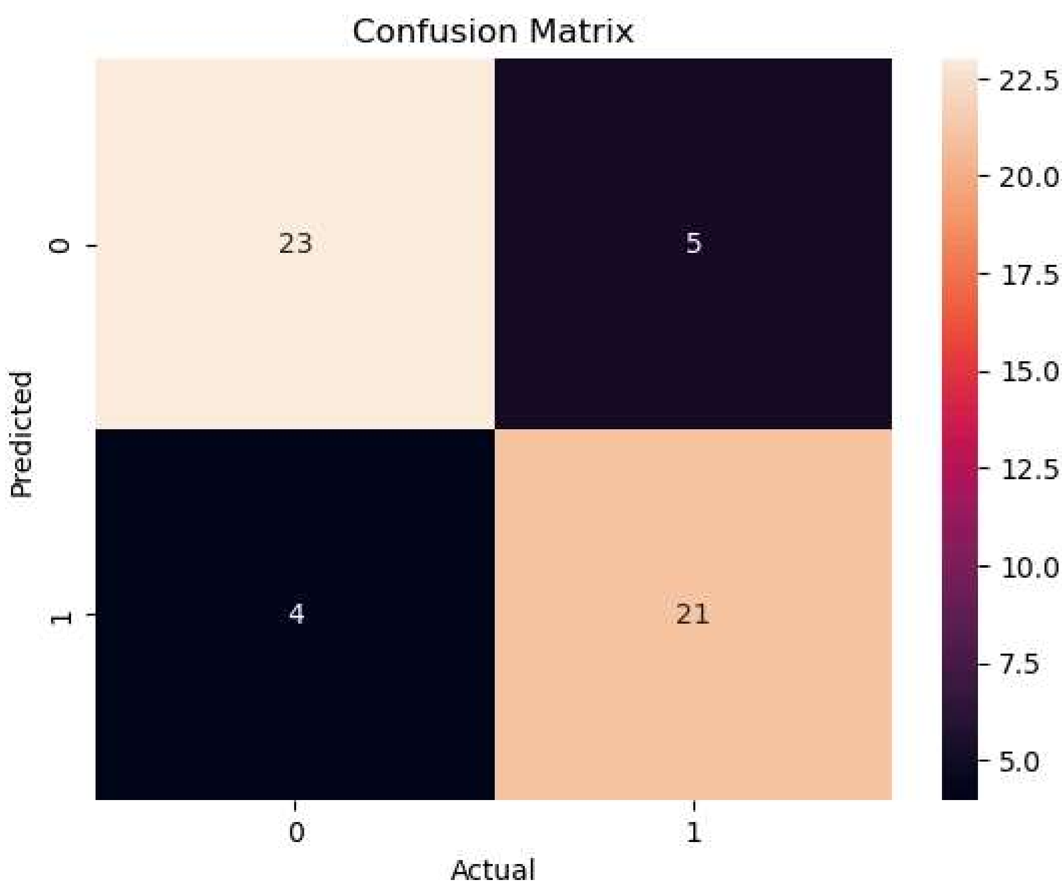
```
80.76923076923077
84.0
83.01886792452831
```

In [115]:

```
print(confusion_matrix(y_true,y_pred))

sns.heatmap(confusion_matrix(y_true,y_pred),annot=True)
plt.title("Confusion Matrix")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
[[23  5]
 [ 4 21]]
```



Conclusion :

```
-The Best Model is Random Forest Classifier.
-It Has Differece Between Training Score & Testing Score is 0.
-Accuracy Score of Random Forest is Higher As Compare To Other Models. i.e  84.90.
```

In [ ]:

```

```