Heart Disease Prediction

Name:Saraansh Chikara

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [6]:
```
df=pd.read_csv("Heart.csv")
```

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

In [53]:
```
df.isnull().sum()
```

Out[53]:
```
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
```

In [112]:
```
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
```

In [113]:
```
df.describe()
```

Out[113]:

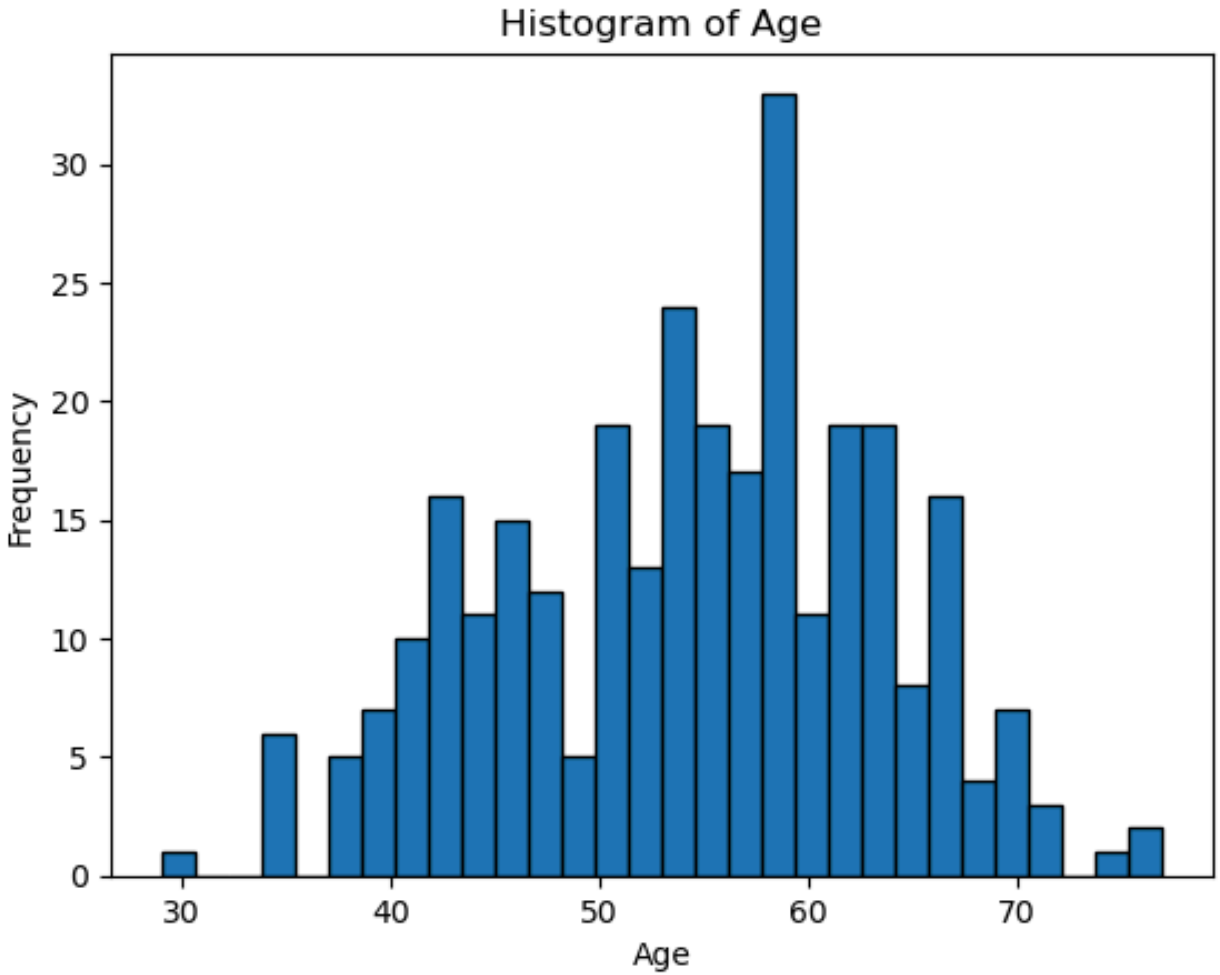
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000

In [27]:
```
a=df.shape
print(f"The data set has {a[0]} rows and {a[1]} columns")
```

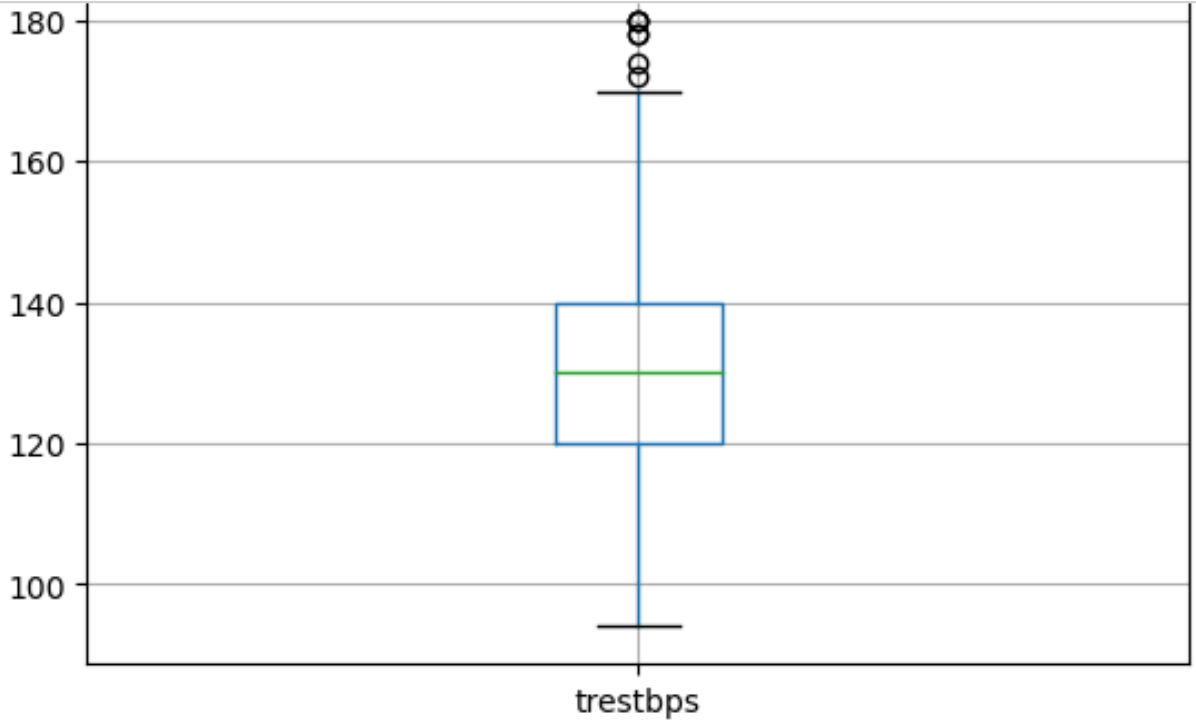
```
The data set has 303 rows and 14 columns
```

Data Visualisation

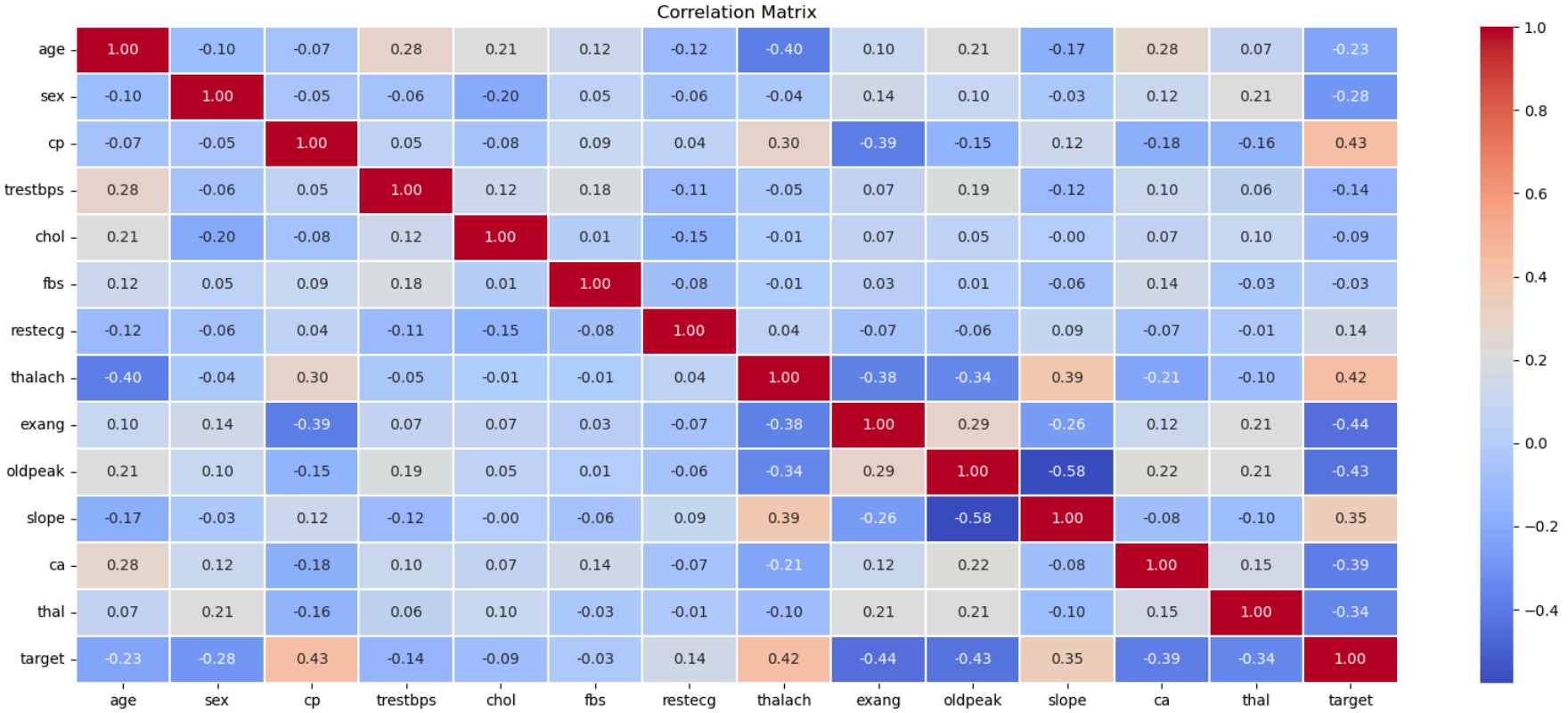
In [67]:
```
plt.hist(df["age"], bins=30,edgecolor="Black")
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



In [8]:
```
df.boxplot(column=['trestbps'])
plt.title('Boxplot of Resting Blood Pressure')
plt.show()
```



In [66]:
```
correlation_matrix = df.corr()
plt.figure(figsize=(20, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.25)
plt.title('Correlation Matrix')
plt.show()
```



Data Modelling

1)Logistic Regression

In [9]:
```
y=df['target']
x=df.drop(columns='target')
```

In [11]:
```
from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test=train_test_split(x,y, test_size=0.30, random_state=40)
```

In [12]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [14]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [15]:
```
model.fit(x_train, y_train)
```

Out[15]:
```
▾ LogisticRegression

LogisticRegression()
```

In [16]:
```
model.intercept_
```

Out[16]:
```
array([0.09066276])
```

In [17]:
```
model.coef_
```

Out[17]:
```
array([[ 0.00210412, -0.75695699,  0.8695811 , -0.41305306, -0.1693742 ,
        -0.0412552 ,  0.38040936,  0.5085677 , -0.27251041, -0.67049861,
         0.28262575, -0.59555475, -0.38458101]])
```

In [18]:
```
y_pred=model.predict(x_train)
print(y_pred)
```

```
[1 1 0 0 0 0 0 1 1 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1
 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1 1 1 0 1 1 0 0 1
 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0 1
 0 1 1 1 0 1 1 0 0 0 1 0 1 0 1 1 1 0 0 1 1 1 1 0 1 0 0 1 0 1 1 1 0 1 0 1 1 1
 1 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1 0]
```

In [19]:
```
y_prediction=model.predict(x_test)
print(y_prediction)
```

```
[1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 0 1 1 1 0
 0 0 1 1 1 1 0 1 1 1 0 0 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
 1 0 1 0 0 1 0 1 0 1 0 1 1 1 0 1 1]
```

In [21]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,recall_score, f1_score
cm=confusion_matrix(y_test, y_prediction)
cm
```

Out[21]:
```
array([[35,  5],
       [ 2, 49]], dtype=int64)
```

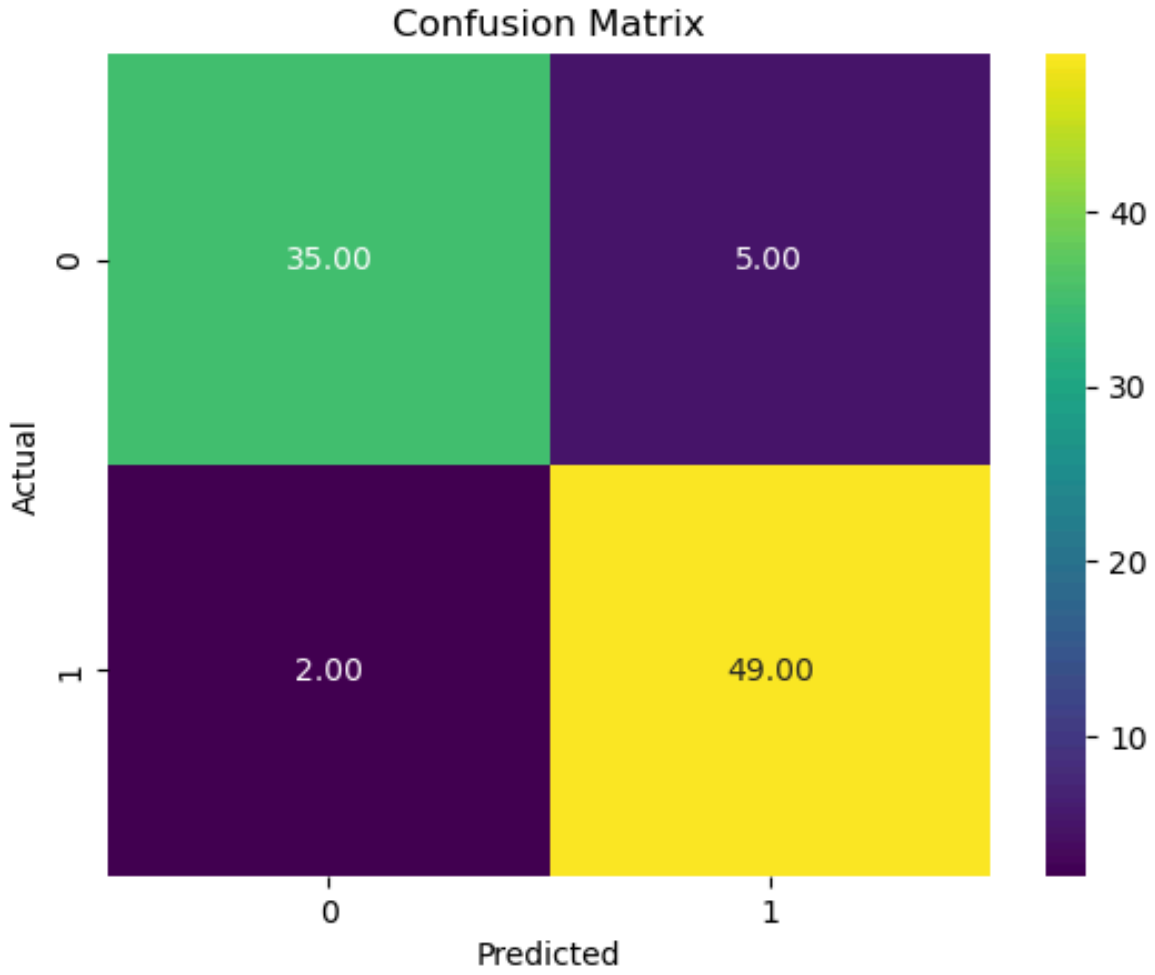
In [22]:
```
accuracy=accuracy_score(y_test, y_prediction)
print(f"Accuracy score is :{accuracy}")
```

```
Accuracy score is :0.9230769230769231
```

In [23]:
```
train_score=model.score(x_train,y_train)
test_score=model.score(x_test,y_test)
print(f"Training score is :{train_score}")
print(f"Testing score is :{test_score}")
```

```
Training score is :0.8349056603773585
Testing score is :0.9230769230769231
```

In [24]:
```
# Plot the confusion matrix as a heatmap
sns.heatmap(cm, annot=True, fmt=".2f", cmap="viridis")
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



In [25]:
```
print(f"Model score :{model.score(x_test,y_test)}")
```

```
Model score :0.9230769230769231
```

In [28]:
```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = model, X = x_train, y = y_train, cv = 5)
print(accuracies)
```

```
[0.74418605 0.76744186 0.80952381 0.88095238 0.73809524]
```

In [29]:
```
print("Accuracy mean: {:.2f} %".format(accuracies.mean()*100))
print("Accuracy max: {:.2f} %".format(accuracies.max()*100))
```

```
Accuracy mean: 78.80 %
Accuracy max: 88.10 %
```

In [65]:
```
a=accuracy_score(y_test, y_prediction)
p=precision_score(y_test, y_prediction)
r=recall_score(y_test, y_prediction)
f1=f1_score(y_test, y_prediction)
print(f"Accuracy score is :{a}")
print(f"Precision score is:{p}")
print(f"Recall score is :{r}")
print(f"F1 score is :{f1}")
```

```
Accuracy score is :0.9230769230769231
Precision score is:0.9074074074074074
Recall score is :0.9607843137254902
F1 score is :0.9333333333333333
```

Decision Tree

In [31]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

In [32]:
```
X = df.drop(columns='target')
Y = df['target']
```

In [33]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

In [34]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [35]:
```
clf = DecisionTreeClassifier()
```

In [36]:
```
clf.fit(X_train, Y_train)
```

Out[36]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [49]:
```
Y_pred=model.predict(X_train)
print(Y_pred)
```

```
[1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0
 0 0 1 1 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0
 1 1 1 1 1 0 1 0 0 0 0 0 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 0 0 0 0 1 1 1
 1 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 0 0 1 1 0 0 1 0 0 0
 0 1 1 1 0 0 1 0 1 1 0 1 0 0 0 1 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0 1 0 0 1 1 0
 1 0 0 0 1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 1 1 0 1 1 1 0 1]
```

In [42]:
```
Y_prediction = clf.predict(X_test)
print(Y_prediction)
```

```
[0 1 1 0 1 1 1 0 0 0 1 0 1 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 0 1 0 0 0 0 1 0 0
 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 0 0 1 0 0 1 1 1 1 1 1 1
 0 1 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1]
```

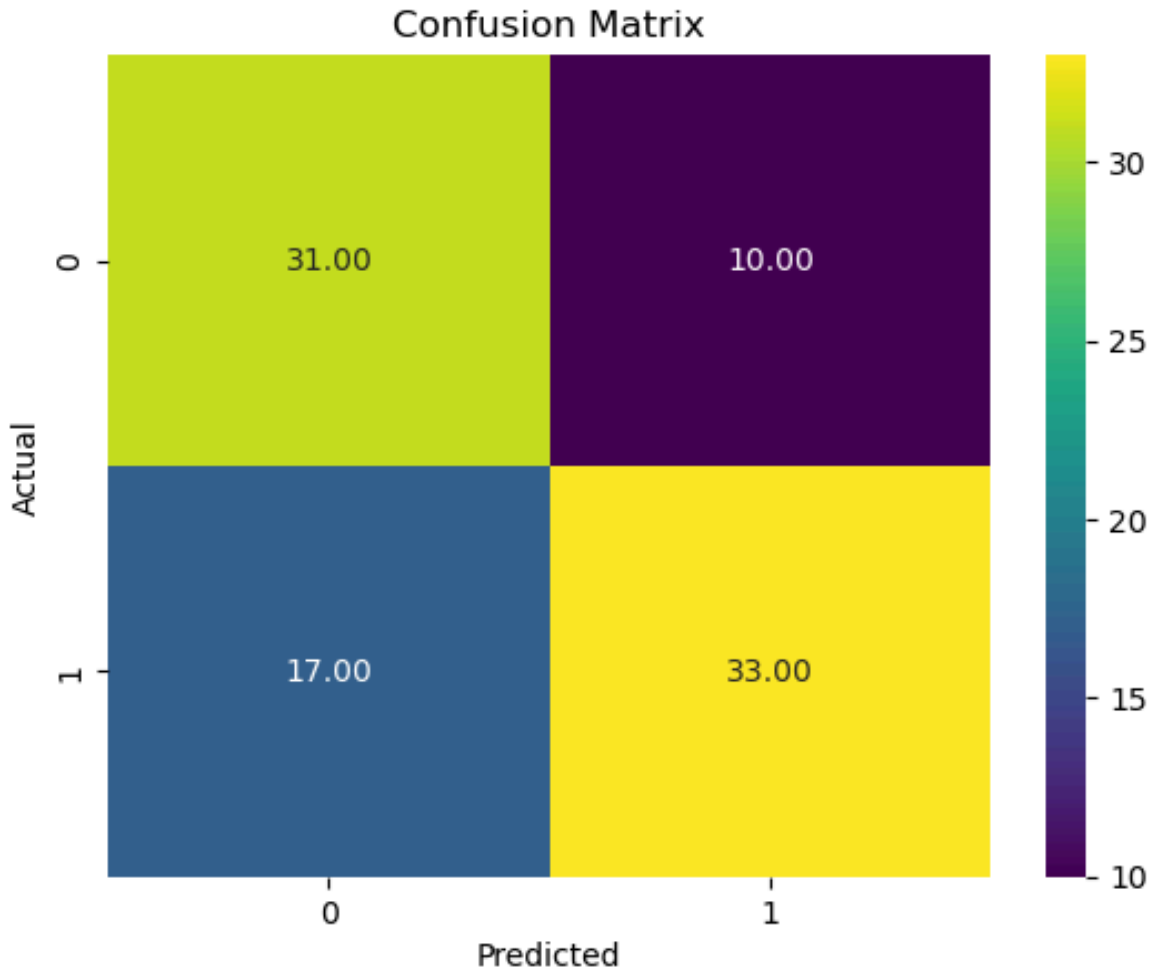
In [52]:
```
train2_score=model.score(X_train,Y_train)
test2_score=model.score(X_test,Y_test)
print(f"Training score is :{train2_score}")
print(f"Testing score is :{test2_score}")
```

```
Training score is :0.8537735849056604
Testing score is :0.8461538461538461
```

In [56]:
```
from sklearn.metrics import confusion_matrix, accuracy_score,  precision_score,recall_score, f1_score
cm=confusion_matrix(Y_test, Y_prediction)
cm
```

Out[56]:
```
array([[31, 10],
       [17, 33]], dtype=int64)
```

In [54]:
```
sns.heatmap(cm, annot=True, fmt=".2f", cmap="viridis")
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



In [55]:
```
print(f"Model score :{model.score(X_test,Y_test)}")
```

```
Model score :0.8461538461538461
```

In [63]:
```
A=accuracy_score(Y_test, Y_prediction)
P=precision_score(Y_test, Y_prediction)
R=recall_score(Y_test, Y_prediction)
F1=f1_score(Y_test, Y_prediction)
print(f"Accuracy score is :{A}")
print(f"Precision score is:{P}")
print(f"Recall score is :{R}")
print(f"F1 score is :{F1}")
```

```
Accuracy score is :0.7032967032967034
Precision score is:0.7674418604651163
Recall score is :0.66
F1 score is :0.7096774193548386
```

In [ ]:
```

```