In [12]:
```
#EXP_8
```

In [13]:
```
#Aim:Logistic Regression
```

In [1]:
```
#Name:Dev Sanjay Vaidya
#Roll no:69
#Sec:B
#Subject:ET-1
#Date:11/09/25
```

To Perform And Analysis Of Logistic Regression Algorithm

Importing The Libraries¶

In [1]:
```
import pandas as pd
import numpy as np
```

Data Acquisitionuing Pandas

In [2]:
```
import os
```

In [3]:
```
os.getcwd()
```

Out[3]:
```
'C:\\Users\\LENOVO'
```

In [4]:
```
os.chdir('C:\\Users\\LENOVO\\Desktop')
```

In [5]:
```
data=pd.read_csv("heart.csv")
```

In [6]:
```
data.head()
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
0	52	1	0	125	212	0	1	168	0	1.0	2
1	53	1	0	140	203	1	0	155	1	3.1	0
2	70	1	0	145	174	0	1	125	1	2.6	0
3	61	1	0	148	203	0	1	161	0	0.0	2
4	62	0	0	138	294	1	1	106	0	1.9	1

In [7]:
```
data.tail()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slo
1020	59	1	1	140	221	0	1	164	1	0.0	
1021	60	1	0	125	258	0	0	141	1	2.8	
1022	47	1	0	110	275	0	0	118	1	1.0	
1023	50	0	0	110	254	0	0	159	0	0.0	
1024	54	1	0	120	188	0	1	113	0	1.4	

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [9]:
```
data.describe()
```

Out[9]:

	age	sex	cp	trestbps	chol	
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.0
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.1
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.3
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.0
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.0
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.0
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.0
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.0

In [10]:
```
data.shape
```

Out[10]: (1025, 14)

In [11]:
```
data.size
```

Out[11]: 14350

In [12]:
```
data.ndim
```

Out[12]: 2

Data preprocessing _ data cleaning _ missing value treatment

In [13]:
```
# check Missing Value by record

data.isna()
```

Out[13]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpea
0	False	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	...	...	
1020	False	False	False	False	False	False	False	False	False	Fals
1021	False	False	False	False	False	False	False	False	False	Fals
1022	False	False	False	False	False	False	False	False	False	Fals
1023	False	False	False	False	False	False	False	False	False	Fals
1024	False	False	False	False	False	False	False	False	False	Fals

1025 rows × 14 columns

In [14]:
```
data.isna().any()
```

Out[14]:
```
age         False
sex         False
cp          False
trestbps    False
chol        False
fbs         False
restecg     False
thalach     False
exang       False
oldpeak     False
slope       False
ca          False
thal        False
target      False
dtype: bool
```

In [15]:
```
data.isna().sum()
```

Out[15]:
```
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
```

Independent and Dependent Variables

In [16]:
```
x=data.drop("target", axis=1)
y=data["target"]
```

Splitting of DataSet into train and Test

In [17]:
```
#splitting the data into training and testing data sets
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2 ,random_sta
```

Logistic Regression

In [18]:
```
import warnings
from sklearn.exceptions import ConvergenceWarning
```

In [19]:
```
from sklearn.linear_model import LogisticRegression
```

In [20]:
```
log = LogisticRegression()
log.fit(x_train, y_train)
```

```
C:\Users\LENOVO\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.p
y:465: 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 in:
    https://scikit-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-regre
ssion
  n_iter_i = _check_optimize_result(
```

Out[20]:

▼ LogisticRegression

LogisticRegression()

In [21]:
```
y_pred1 = log.predict(x_test)
```

In [22]:
```
from sklearn.metrics import accuracy_score
```

In [23]:
```
accuracy_score (y_test,y_pred1)
```

Out[23]: 0.7853658536585366

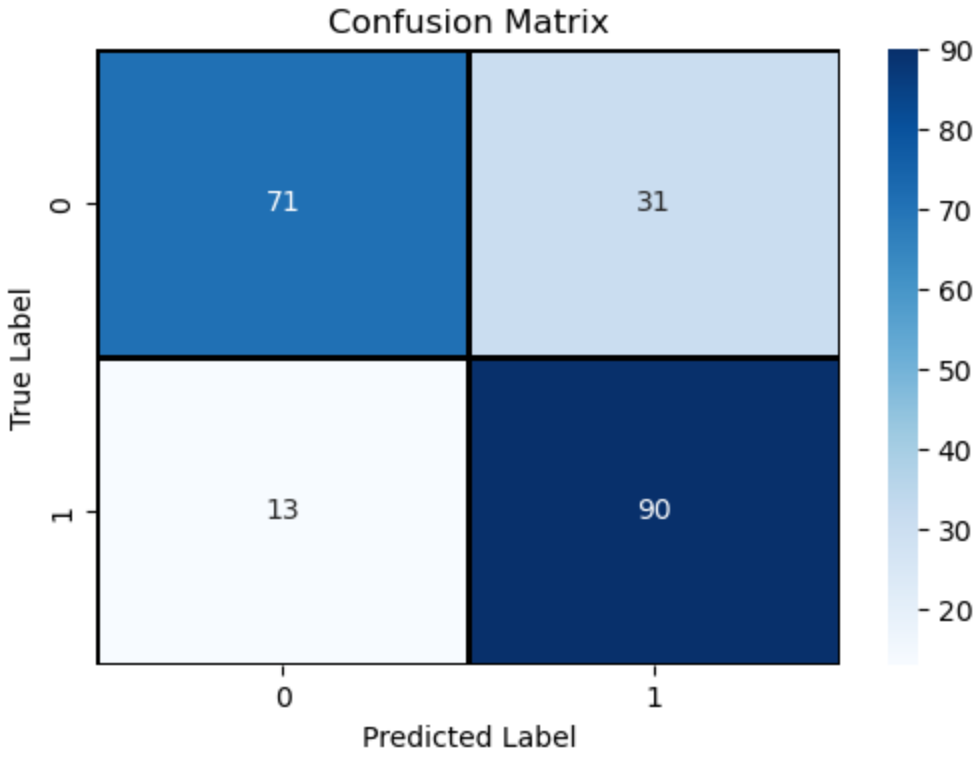
In [24]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

In [25]:
```
cm = confusion_matrix(y_test, y_pred1)
```

In [26]:
```
labels = np.unique(y_test)  # Get unique class labels
cm_df = pd.DataFrame(cm, index=labels, columns=labels)
```

In [27]:
```
# Plot confusion matrix using seaborn
plt.figure(figsize=(6, 4))
sns.heatmap(cm_df, annot=True, fmt='d', cmap='Blues', linewidths=1, linecolc

plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```


Confusion Matrix
71
31
13
90
0
1
True Label
0
1
Predicted Label
90
80
70
60
50
40
30
20

In [ ]: