In [24]:
```
#Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [62]:
```
#Importing dataset
df = pd.read_csv("heart.csv")
df
```

Out[62]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 14 columns

In [63]:
```
df.shape
```

Out[63]:
```
(303, 14)
```

In [64]:
```
df.describe()
```

Out[64]:

	age	sex	cp	trestbps	chol	fbs	restecg
count	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
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000

In [65]:
```
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 [66]:
```
df.isnull().sum()
```

Out[66]:
```
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 [97]:
```
#Seperating features and target
X = df.drop(columns='target', axis=1)
y = df['target']
```

In [98]:
```
#Train test split
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_state=
```

In [99]:
```
#Performing scalarization on training data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
sc_X_train = scaler.fit_transform(X_train)
sc_X_test = scaler.transform(X_test)
```

In [100…
```
#Performing Logistic Regression with L1 (Lasso)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(penalty='l1', solver='saga', max_iter=1000)
model.fit(sc_X_train, y_train)
```

Out[100…
```
LogisticRegression
LogisticRegression(max_iter=1000, penalty='l1', solver='saga')
```

In [101…
```
y_pred = model.predict(sc_X_test)
```

In [102…
```
#Evaluation Metrics
from sklearn.metrics import accuracy_score, confusion_matrix
print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
0.8688524590163934
[[25  7]
 [ 1 28]]
```

In [103…
```
#Performing Logistic Regression with L2 (Ridge)
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression(penalty='l2', solver='liblinear', max_iter=1000)
model.fit(sc_X_train, y_train)
```

Out[103…

```
LogisticRegression
LogisticRegression(max_iter=1000, solver='liblinear')
```

In [104…

```
y_pred = model.predict(sc_X_test)
```

In [105…

```
#Evaluation Metrics
from sklearn.metrics import accuracy_score, confusion_matrix
print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
0.8688524590163934
[[25  7]
 [ 1 28]]
```

In [106…

```
#Performing Logistic Regression with (Elastic Net)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(penalty='elasticnet', l1_ratio=0.5 ,solver='saga', max_i
model.fit(sc_X_train, y_train)
```

Out[106…

```
LogisticRegression
LogisticRegression(l1_ratio=0.5, max_iter=1000, penalty='elasticnet',
                   solver='saga')
```

In [107…

```
y_pred = model.predict(sc_X_test)
```

In [110…

```
#Evaluation Metrics
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
0.8688524590163934
[[25  7]
 [ 1 28]]
```

In [ ]: